Fachhochschule Aachen
Campus Jülich

Bachelorarbeit

Erhebung von Messdaten zur Usability-Evaluation des Informationsportals JuLib eXtended

vorgelegt von Philipp Pollack

Jülich, den 7. August 2014

Fachbereich:	Medizintechnik und Technomathematik
Studiengang:	Scientific Programming
Matrikelnummer:	855489

Erstellt in der

Zentralbibliothek

des

Forschungszentrums Jülich

Diese Arbeit wurde betreut von:

1. Prüfer: Prof. Dr. rer. nat. Volker Sander
2. Prüfer: Dipl. Phys. Ing. Waldemar Hinz

Eigenständigkeitserklärung

Diese Arbeit ist von mir selbstständig angefertigt und verfasst. Es sind keine anderen als die angegebenen Quellen und Hilfsmittel benutzt worden.

Ort, Datum

Philipp Pollack

Zusammenfassung

Das Informationsportal JuLib eXtended ist ein zentraler Dienst der Zentralbibliothek des Forschungszentrums Jülich. Benutzt wird JuLib eXtended über eine Weboberfläche. Über diese können die Mitarbeiter des Forschungszentrums im Bestand nach Büchern, E-Books oder Zeitschriften suchen und weitere Funktionen nutzen, wie zum Beispiel die Literaturbestellung bzw. Verlängerung der Rückgabefrist von Medien.

Da viele Nutzer aus den verschiedensten Fachrichtungen JuLib eXtended verwenden, ist ein wichtiges Kriterium die Nutzbarkeit des Dienstes und welche Anforderungen die Anwender an die Bedienung stellen. Die Usability-Evaluation ist ein Werkzeug, um eine Benutzeroberfläche in Hinblick auf diese Kriterien zu untersuchen und daraufhin zu optimieren. Zu diesem Zweck gibt es mehrere Methoden, die sich in Art und Weise des Vorgehens unterscheiden. Zum einen gibt es Unterschiede bei der allgemeinen Durchführung der Evaluation, beispielsweise welche Personenkreise zu Rate gezogen werden oder wie der Ablauf geplant wird. Aber auch die betrachteten Schwerpunkte können sich je nach verwendeter Methode voneinander abheben, so könnte zum Beispiel die Benutzbarkeit für neue Anwender im Fokus stehen.

Im Rahmen dieser Arbeit sollen die Grundlagen für eine Usability-Evaluation von JuLib eXtended gelegt werden. Zunächst werden gängige Verfahren und einige praktische Anwendungen dazu betrachtet. Darauf basierend wird dann ein eigenes Verfahren, speziell für die Anforderungen der Zentralbibliothek, entworfen. Abschließend wird eine Web-Anwendung entwickelt, um Daten von Probanden zu erheben und diese in einer Datenbank zu speichern. Ziel ist somit ein eigenes Verfahren samt Testumgebung zu schaffen, um eine zu späterem Zeitpunkt erfolgende Auswertung zu vereinfachen, indem die Datenerhebung bereits standardisiert und weitgehend automatisiert geschieht.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Weltweit gibt es unzählige Webseiten und Webapplikationen. Allesamt bieten diese den Nutzern verschiedenste Dienste an und konkurrieren bei gleichen Leistungen untereinander. Nur Seiten, die die Nutzer zufriedenstellen, haben eine Chance über eine längere Zeitspanne zu bestehen. Wohingegen Webseiten, deren Bedienung und Aussehen für den Nutzer unzulänglich sind, schnell wieder vom Markt genommen werden, da ihre Bereitstellung nicht rentabel ist. Wichtigstes Mittel für Betreiber, um ihre Webseiten auf die Nutzbarkeit zu überprüfen, sind Usability-Evaluationen. Bei diesen wird nach festgelegten Kriterien jedes Detail des Webauftritts bewertet und notiert, sodass im Anschluss eine Analyse zur Verbesserung oder ggf. Neugestaltung der Seite möglich ist. Danach kann dann in regelmäßigen Abständen eine erneute Evaluation stattfinden, um Änderungen zu überprüfen oder auf eventuelle neue Anforderungen zu reagieren.

Die Zentralbibliothek des Forschungszentrums Jülich betreibt das Informationsportal JuLib. Über dieses können die Mitarbeiter des Forschungszentrums im Bestand nach Büchern, E-Books oder Zeitschriften suchen und weitere Funktionen nutzen, wie zum Beispiel die Literaturbestellung oder Verlängerung der Rückgabefrist von Medien.

Bisher besteht für die Nutzer von JuLib die Möglichkeit, über eine simple E-Mail ihre Meinung der Zentralbibliothek zukommen zu lassen. Doch um den Ansprüchen der vielen Mitarbeiter gerecht zu werden, sollen zukünftig Usability-Evaluationen bezüglich JuLib durchgeführt werden.

Ziel dieser Arbeit ist es daher auf der einen Seite ein Verfahren passend zu den Anforderungen der Zentralbibliothek zu entwickeln und auf der anderen Seite anschließend eine Software zur Usability-Evaluation von JuLib zu entwickeln, die dieses Verfahren umsetzt. Von dieser Arbeit nicht näher beleuchtet bleibt die Auswertung der erhobenen Daten, sondern ausdrücklich nur die Erhebung eben jener.

In Kapitel 2 wird zunächst erläutert, was genau unter einer Usability-Evaluation zu verstehen ist. Danach werden dann exemplarisch einige Verfahren für eine solche Evaluation dargestellt. Daraufhin wird dann in Kapitel 3 detailliert das Verfahren, wie es in der

Zentralbibliothek zur Anwendung kommen könnte, erläutert. Probanden sollen ihnen gestellte Aufgaben JuLib betreffend ausführen. Alle getätigten Aktionen sollen protokolliert und anhand des Nutzerverhaltens mögliche Probleme entdeckt werden. Die anschließenden Kapitel beschäftigen sich mit der Umsetzung einer Evaluations-Umgebung.

Kapitel 4 nennt die dafür nötigen Grundlagen, die Speicherung der anfallenden Daten wird in Kapitel 5 beschrieben.

Danach erläutert Kapitel 6 detailliert die eigentliche Webanwendung zur Durchführung einer Usability-Evaluation.

Abschließend wird in Kapitel 7 ein Fazit gezogen und zusätzlich gibt es einen Ausblick auf mögliche Erweiterungen und eine Auswertung der Evaluation.

Kapitel 2

Usability-Evaluation

Was ist eigentlich eine Usability-Evaluation? Und wie wird diese dann in der Praxis durchgeführt? Diese beiden Fragen bilden den Schwerpunkt dieses Kapitels. Auch wenn in vielen Branchen bereits seit geraumer Zeit solche Evaluationen durchgeführt werden, soll an dieser Stelle nochmal der grundlegende Aufbau erklärt werden. Dazu wird zunächst eine allgemeine Definition gegeben, um danach anhand einiger Beispiele theoretische Verfahren und praktische Anwendungen aufzuzeigen.

2.1 Definition

Bei einer Usability-Evaluation [Wik] wird eine grafische Benutzeroberfläche unter bestimmten Kriterien auf ihre Nutzbarkeit hin untersucht. Dazu werden am Anfang bestimmte Kriterien aufgestellt, welche von dem Produkt erfüllt werden sollen. Als Ergebnis liefert eine Usability-Evaluation Probleme und je nach Methode eventuell direkt mögliche Lösungen für diese. Alle gefundene Probleme können dann im Anschluss mit einem Severity-Rating[1] nach Schwere eingeteilt und an die zuständigen Stellen weitergegeben werden. Sollten sich die Anforderungen oder das untersuchte Produkt ändern, wird die Evaluation wiederholt, um wie bei einer Rekursion das Produkt kontinuierlich zu verbessern.

In der DIN EN ISO 9241-11 [ISO99] sind drei Kriterien für die Usability einer Software festgelegt: Effektivität, Effizienz und Zufriedenstellung. Effektivität sagt dabei nur aus, dass die zu erledigenden Aufgaben vollständig und korrekt ausgeführt werden. Dies wird vom Kriterium der Effizienz erweitert, indem der Aufwand hinzugezogen wird. Dieser sollte bei gleichzeitiger Zuverlässigkeit möglichst gering sein. Abschließend beschreibt das dritte Kriterium die Zufriedenheit der Benutzer. Diese ist allerdings subjektiv und abhängig von der Zielgruppe und muss dahingehend angepasst werden. Alle drei Kriterien

[1] http://www.stcsig.org/usability/newsletter/9904-severity-scale.html

sollten von einem umfassenden Test erfasst werden, wobei eine Evaluation aber auch gezielt einzelne Schwerpunkte setzen kann.

Um eine Usability-Evaluation durchzuführen, gibt es mehrere Verfahren/Methoden. Diese sind dabei in verschiedene Typen unterteilt, die bei manchen Gegebenheiten Vor- und Nachteile haben. Eine erste Unterteilung kann in formative und summative Verfahren vorgenommen werden. Formative Methoden finden Anwendung während der Entwicklung der Oberfläche sobald ein erster Entwurf erstellt wurde. Sie bieten konkrete Verbesserungsvorschläge, indem sie qualitative Daten zur Identifikation von Problemen nutzen. Summative Verfahren hingegen kommen nach Abschluss der Entwicklung zum Einsatz. Hier werden statt qualitativer Daten bei formativen Verfahren quantitative Daten erhoben, wie beispielsweise Zeiten und Anzahl der Fehler/ Versuche, die objektiv gemessen werden können. Sie stellen einen Vergleich zwischen festgelegtem Ziel und momentan umgesetztem Produkt an, während formative Verfahren auf eine Korrektur der laufenden Entwicklungsschritte abzielt.

Auch bei der Durchführung der eigentlichen Evaluation gibt es unterschiedliche Ansätze. Diese kann man in die beiden Bereiche expertenorientiert/analytisch und benutzerorientiert/empirisch unterteilen. Analytische Verfahren zeichnen sich vor allem durch feste Schemata aus, nach denen eine Evaluation durchgeführt wird, während dies bei empirischen Verfahren nur eingeschränkt gilt. Viel mehr muss bei empirischen Verfahren für jede Evaluation eine eigene Vorgehensweise entwickelt werden, die speziell an die jeweiligen Umstände angepasst wird. In den folgenden Abschnitten werden daher einige Verfahrensweisen aus beiden Bereichen aufgezeigt.

2.2 Analytische Methoden

Analytische Verfahren werden von Usability-Experten durchgeführt, daher heißen sie auch expertenorientierte Verfahren. Echte Nutzer des Produkts sind nicht involviert, statt dessen wird das Produkt systematisch von Experten untersucht. Die Experten sollten dabei generell unabhängig sein und nicht z.B. zum Entwicklerteam gehören, um ein möglichst unvoreingenommenes Ergebnis zu ermöglichen. Solche Methoden können relativ schnell und flexibel durchgeführt werden. Allerdings sind externe Experten gegebenenfalls mit teils hohen Kosten verbunden. Im Folgenden werden kurz ein paar gängige Methoden vorgestellt.

2.2.1 Heuristische Evaluation

Bei der heuristischen Evaluation [Wil14, 1-32] werden, wie der Name schon sagt, Heuristiken[2] eingesetzt, um das zu überprüfende Produkt zu bewerten. Sie gehört zu der Gruppe der formativen und expertenorientierten Verfahren, das heißt, dass sie vor der Fertigstellung von Experten durchgeführt wird. Somit wird die Evaluation im Schnitt von 2-10 Prüfern und einem Leiter durchgeführt. Der Leiter plant die gesamte Evaluation: Er wählt die Heuristiken aus, stellt das Team zusammen, instruiert die Prüfer und fasst am Ende die Einzelergebnisse zusammen. Die Hauptaufgabe ist dabei die Auswahl der Heuristiken. Diese müssen dem Anspruch der Evaluation angemessen gewählt werden. Generell bezeichnet Heuristik ein Verfahren, bei dem mit begrenztem Wissen eine möglichst gute Lösung erzielt wird. Dies bedeutet in diesem Kontext, dass die Heuristiken Regeln festlegen, mithilfe derer eine Einschätzung des Produkts möglich ist. Heuristiken sollten für die Prüfer leicht verständlich, relevant für das Produkt und leicht zu merken sein. Dabei können verschiedene Ansätze verfolgt werden. Einer ist der objektbasierte Ansatz, bei dem einzelne Elemente der Benutzeroberfläche betrachtet werden, wie beispielsweise Fehlermeldungen oder Eingabemasken. Im Gegensatz dazu werden beim aufgabenbasierten Ansatz den Prüfern Aufgaben gestellt, welche diese dann bearbeiten sollen. Diese beiden Ansätze gibt es auch als hybride Ausführung, bei welcher zunächst Aufgaben gelöst und danach mit denselben Heuristiken einzelne Elemente betrachtet werden.

Der größte Vorteil dieser Art der Evaluation ist, dass sie besonders einfach durchzuführen ist. Sie kann zu jeder Zeit während der Entwicklung angesetzt werden und erfordert keine Koordination unter den Prüfern. So können alle für sich selbst mit den gegebenen Heuristiken das Produkt überprüfen. Auf der anderen Seite birgt aber genau dieses Vorgehen auch Probleme. Durch das getrennte Arbeiten ist die gefundene Menge an Problemen sehr subjektiv und nicht einheitlich, d.h. was ein Prüfer als Problem identifiziert, hält ein anderer für unproblematisch. Auch die Genauigkeit, mit der Probleme beschrieben werden oder mit der generell gesucht wird, ist unterschiedlich. Heuristische Evaluationen neigen dazu, meist nur sehr allgemeine und oberflächliche Probleme aufzudecken.

Dies alles erfordert vom Leiter der Evaluation ein hohes Maß an Geschick, um eine vollständige Liste aller Probleme zu erstellen.

[2] http://www.wissen.de/lexikon/heuristik

2.2.2 Perspective-Based UI Inspection

Bei dieser Methode kommen verschiedene Perspektiven zum Einsatz. Durchgeführt wird eine Perspective-Based UI Inspection [Wil14, 49-63] von 2-10 Usability-Experten und einer Person als Leiter. Die Aufgaben des Leiters sind ähnlich der heuristischen Evaluation, außer, dass er hier keine Heuristiken auswählen, sondern eine Menge von Perspektiven zusammenstellen muss. Jeder Prüfer bekommt dann eine feste Perspektive zugewiesen, aus welcher er das Produkt betrachtet. Die genaue Auswahl der Perspektiven ist eine schwierige Angelegenheit und abhängig von den zu erwartenden Nutzern (Zielgruppe) des Produktes, sowie den Aspekten, die untersucht werden sollen. Als Beispiel sei die Nutzbarkeit für Behinderte genannt, wo die Perspektive eines Sehbehinderten vorstellbar ist. Auch Hintergründe, Erfahrungen und Personenbeschreibungen gehören zur Ausgestaltung einer Perspektive dazu.

Eine Perspective-Based UI Inspection kann zu jeder Zeit während der Entwicklung angesetzt werden, es muss nur eine Repräsentation der Oberfläche verfügbar sein. Sie gehört zu den formativen und expertenorientierten Methoden und bietet sich vor allem an, wenn kein Zugriff auf reale Nutzer möglich ist, aber das Produkt später mal von einer breiten Masse an Menschen genutzt wird. So kann das Produkt von mehreren Standpunkten aus betrachtet werden, was allerdings auch eine Herausforderung für die Tester darstellen kann, wenn die Perspektiven schwierig umzusetzen sind. Es ist beispielsweise für einen Experten in dem jeweiligen Gebiet schwierig, sich in die Rolle eines neuen unversierten Nutzers zu versetzen. Um dies für die Tester zu erleichtern, sollten die Rollenbeschreibungen ausführlich sein und keine sehr komplexen Anforderungen stellen. Schlussendlich lässt sich aber nicht überprüfen, ob der Tester wirklich strikt seine Rolle beachtet hat. Generell gelten hier auch die Vor- und Nachteile einer heuristischen Evaluation.

Erweitern lässt sich die Perspective-Based UI Inspection unter anderem durch die Verwendung von Kriterien-Katalogen. Dabei wird die eigentliche Untersuchung in mehrere Einheiten geteilt. Jede Einheit zielt dann nur noch auf ein bestimmtes Kriterium ab, so hat dann jede Einheit auch ihre eigenen spezialisierten Perspektiven. Durch diese Erweiterung ist es möglich einen detaillierteren Überblick zu gewinnen.

2.2.3 Pluralistic Usability Walkthrough

Diese Methode unterscheidet sich in ihrer Art von den vorigen Beiden. Der Pluralistic Usability Walkthrough [Wil14, 81-97] kann nicht eindeutig zu einer bestimmten Art gezählt werden. So kann der Walkthrough bereits in der Entwicklung, aber auch nach deren Abschluss eingesetzt werden. Bei der Durchführung sind neben Experten auch Nutzer des

Produktes beteiligt. Bei dieser Methode werden zunächst vom Leiter eine Menge an Aufgaben festgelegt, und alle Mitglieder der Gruppe lösen diese für sich allein. Danach treffen sich alle Mitglieder, um über ihre Lösungswege und die dabei aufgetretenen Probleme zu diskutieren.

Aufgrund dieses einfachen Aufbaus hat der Pluralistic Usability Walkthrough einige Stärken. Da reale Nutzer und nicht nur Experten involviert sind, können auch repräsentative Meinungen von den Endkunden berücksichtigt werden. Indem die Experten noch aus verschiedenen Fachgebieten stammen, deckt man somit ein breites Spektrum an Perspektiven ab, ohne explizit Rollen zuordnen zu müssen. Durch die Diskussionsphase ist es auch möglich, neben den Problemen auch Lösungen zu finden. Die besten Ergebnisse lassen sich dabei mit linearen Aufgaben erzielen, wobei es auf der anderen Seite aber auch zu Problemen bei der Durchführung kommen kann. So sind größere beziehungsweise komplexere Aufgaben dafür meist ungeeignet. Auch liegt der Schwerpunkt aufgrund der Fokussierung bei der Untersuchung von Lernschwierigkeiten und nicht bei der Effizienz der Nutzung. Zudem ist ein „starker“ Leiter von Nöten, um die Diskussion gezielt zu steuern, damit diese sich nicht in Nebensächlichkeiten verläuft.

2.3 Empirische Methoden

Empirische Methoden setzen im Gegensatz zu analytischen Verfahren auf die Beteiligung von echten Nutzern, indem diese bei der Benutzung beobachtet werden. Sie bilden die Zielgruppe des Produkts und so können direkt Rückschlüsse auf die Nutzbarkeit gezogen werden. Diese Methoden unterliegen strikten Anforderungen, da der komplette Testablauf festgelegt sein muss, bevor die Nutzer hinzugezogen werden können. So muss eine gewisse Planungsphase beachtet werden. Empirische Methoden sind meist kostengünstiger durchzuführen, da einem, im Gegensatz zu Experten, oft wirkliche Nutzer zur Verfügung stehen. Bei dieser Art von Evaluationen gibt es wenige allgemeine Vorgehen[3], aber im Folgenden sollen zwei Möglichkeiten aufgezeigt werden.

2.3.1 Umfrage

Eine Umfrage [Umf] stellt eine der einfachsten Varianten, eine Usability-Evaluation durchzuführen, dar, um eine große Anzahl Personen zu erreichen. Bei einer Umfrage wird zur Erfassung der Meinungen und Erfahrungen ein Fragebogen erstellt. Ein solcher Fragebogen muss sich nicht gezielt an wirkliche Nutzer richten, sondern kann sogar teilweise von

[3] http://usability-toolkit.de/usability/usability-methoden/

Personen ausgefüllt werden, die das Produkt noch niemals benutzt haben. Auch ist eine räumliche Anwesenheit nicht erforderlich, da ein Fragebogen auch über eine Onlineplattform bereitgestellt werden kann.

Während die eigentliche Durchführung der Umfrage keine sonderlichen Hürden bereitet, muss diese sehr wohl gut vorbereitet sein, um zufriedenstellende Ergebnisse zu liefern. Für die Auswahl des Fragebogens bieten sich grundsätzlich zwei Möglichkeiten an. Zum einen kann ein vorgefertigter und standardisierter Fragebogen genommen werden. Zum anderen kann ein eigener Fragebogen entworfen werden, der besser auf Eigenheiten des eigenen Produkts eingehen kann. Dagegen bietet die standardisierte Variante eine bessere Vergleichbarkeit. In jedem Fall spielt die Zielgruppe eine wichtige Rolle: Entweder muss diese im Vorfeld der Umfrage festgelegt werden, oder die Umfrage hat gerade das Ziel die Zielgruppe zu bestimmen.

Eine Auswertung der Umfrage kann unter anderem mithilfe statistischer Werkzeuge erfolgen, um Mittelwerte und Abweichungen zu bestimmen.

2.3.2 Lautes Denken

Lautes Denken [TU] oder auch Thinking Aloud ist ein Verfahren dessen Ursprung aus der Kognitionspsychologie[4] kommt. Heute wird dieses in verschiedensten Bereichen eingesetzt, wie auch bei Usability-Evaluationen.

Bei der Methode bekommen Probanden mehrere Aufgaben zur Bearbeitung gestellt. Dabei sollen die Probanden ihre Gedanken, Gefühle, Absichten sowie Erwartungen laut aussprechen. Ein Beobachter protokolliert alles, um eine spätere Auswertung zu ermöglichen. Aufgrund des Versuchsaufbaus bieten sich Einzelsitzungen in einer Art Labor als Umgebung an, um den Probanden von äußeren Einflüssen abzuschotten und eine bessere Beobachtung zu ermöglichen. Oftmals kommen neben dem menschlichen Beobachter weitere Systeme, wie Video- oder Tonaufzeichnung, hinzu, um den Beobachter zu entlasten.

Es gibt hauptsächlich zwei Varianten des Thinking Aloud. Die erste ist das Concurrent Thinking Aloud, das klassische Szenario. Während der Proband die Aufgaben bearbeitet, verbalisiert er unmittelbar seine Gedanken etc., welche von einem Beobachter notiert werden. Das Feedback zur aktuellen Situation kann so direkt mit dem Verhalten abgeglichen und entsprechend kategorisiert werden. Auch Defizite können somit sofort gefunden werden, ohne dass der Nutzer unterbrochen wird oder nochmals rekapitulieren muss. Allerdings bedeutet das gleichzeitige Formulieren seiner Gedanken auch eine gewisse Ablenkung, sodass von einem negativen Einfluss auf die Aufgabenperformanz ausgegangen

[4] https://de.wikipedia.org/wiki/Kognitionspsychologie

werden muss, welcher besonders bei z.B. Feldstudien oder Leistungsmessungen unerwünscht ist.

Die andere Variante ist die des Retrospective Thinking Aloud. Im Gegensatz zur ersten Variante bearbeitet der Proband hierbei die Aufgaben ungestört. Erst nachdem er mit allen Aufgaben fertig ist, muss er seine Gedanken rückblickend äußern. Durch eine strukturierte Fragestellung seitens des Beobachters ist eine bessere Generalisierung und Vergleichbarkeit einzelner Tests möglich. Auch kann der Proband seine Gedanken in Hinblick auf das Gesamtsystem äußern und nicht nur beschränkt auf einen Teilaspekt. Bei dieser Variante sind auch Leistungsmessungen möglich, da der Proband sich komplett auf die Aufgabe konzentrieren kann. Allerdings neigt ein nachträgliches Feedback zu Ungenauigkeiten in Bezug auf Details, da der Proband sich vordergründig an die stärksten Eindrücke erinnern wird. Auch muss hier ein größerer Zeitaufwand beachtet werden, denn neben dem reinen Test muss auch die Zeit für die anschließende Befragung veranschlagt werden.

2.4 Anwendungsbeispiele

Um nicht nur Usability-Evaluation anhand von theoretischen Methoden vorzustellen, widmet sich dieser Abschnitt der Vorstellung von praktisch durchgeführten Evaluationen. Nur mithilfe solcher kann ein umfangreicher Eindruck entstehen und aufgezeigt werden, wie die Anwendung ausschaut.

2.4.1 RealEYES Projekt

Das RealEYES Projekt[OHE01] der Universität Rostock[5] in Zusammenarbeit mit dem Fraunhofer IGD[6] ist ein weiteres anschauliches Beispiel für eine empirische Usability-Evaluation. Die Evaluation wurde im Kontext des Onlineshoppings durchgeführt, indem die Probanden typische Aufgaben in einem Webshop erledigen sollten.

Bei diesem Projekt wurde ein System zur Erfassung der Augenbewegung genutzt, um aus dieser den im Fokus stehenden Teil der Webseite zu bestimmen. Dabei wurden vor allem drei Aspekte betrachtet: Als erstes der Zusammenhang zwischen Augenbewegung und Mauszeiger, des Weiteren das Verhalten der Augen während der Erfassung von neuen Inhalten oder bei Wartezeiten und zuletzt das Verhältnis zwischen textuellen und nichttextuellen Inhalten, die aktiv betrachtet wurden.

[5] http://www.uni-rostock.de/
[6] https://www.igd.fraunhofer.de/

Das System bestand aus einer Infrarot-Kamera, welche die Pupillen filmte und einem Kontroll-PC, der daraus die Augenbewegung in Koordinaten auf dem Bildschirm umrechnete. Der Proband musste während des Testes seinen Kopf auf einer augenärztlichen Kopfstütze ablegen, damit das System präzise arbeiten konnte. Zusätzlich zur Blickregistrierung wurde Thinking Aloud in Verbindung mit herkömmlicher Video- und Tonaufzeichnung eingesetzt.

2.4.2 BibEval – Ein Kriterienkatalog zur Usability-Evaluation

Ein weiterer Ansatz für ein empirisches Verfahren ist der Kriterienkatalog BibEval [WtB11] des Schweizerischen Instituts für Informationswissenschaft. Entwickelt wurde dieser im Rahmen des Innovationsprojekts „E-lib.ch: Elektronische Bibliothek Schweiz“ und soll für andere Bibliotheken als Grundlage für eine eigenständige Usability-Evaluation der eigenen Webanwendungen dienen. Daher wurde der Kriterienkatalog modular aufgebaut, um von einer umfassenden Analyse aller Online-Angebote bis hin zu einem Test von einzelnen Komponenten alle Anwendungsfälle abdecken zu können. Im Kern ist BibEval ein Fragenkatalog der auf einer hierarchisch strukturierten Liste basiert. Diese Liste ist in die drei Stufen „Sektionen“, „Subsektionen“ und „Komponenten“ unterteilt. Die verschiedenen Angebote des Webauftritts einer Bibliothek werden somit auf die Bereiche einsortiert, und es wird ermöglicht zu jeder Komponente Fragen zu stellen.

Die eigentliche Evaluation wird mithilfe einer interaktiven Weboberfläche durchgeführt. Auf dieser kann der Tester die für ihn interessanten Bereiche auswählen und muss danach die dazugehörigen Fragen abarbeiten. Diese Fragen werden allerdings nicht per freiem Text, sondern mittels Auswahlmöglichkeiten beantwortet. Diese nehmen eine Einteilung von Problemen in verschiedene Kategorien vor, welche von „Kein Usability-Problem“, über „Gravierendes Usability-Problem“ bis zu „Feature nicht umgesetzt, obwohl notwendig.“ reichen. Zusätzlich kann aber dennoch zu jeder Frage ein Kommentar abgegeben werden. Abschließend kann der Nutzer den ausgefüllten Fragebogen als CSV- oder PDF-Datei exportierten.

2.4.3 Evaluation der Literatursuchmaschine der Universität Konstanz

An der Universität Konstanz wurde im Zuge der Einführung der neuen Literatursuchmaschine KonSearch eine Usability-Evaluation von eben dieser durchgeführt [Luc11]. Bei der Evaluation wurde nicht nur ein Verfahren genutzt, sondern eine Reihe verschiedener Methoden. Dazu wurden einzelne Arbeitsgruppen gebildet, die unabhängig voneinander jeweils eine Methode umsetzten. Zum Einsatz kamen eine Benutzerbefragung, ein

summativer User-Test, ein formativer User-Test sowie Eye-Tracking. Für alle Verfahren wurden Studierende der Universität als Probanden eingesetzt.

Bei der Benutzerbefragung wurden zunächst von dem Team intern Anforderungen an KonSearch formuliert. Diese wurden mithilfe kleiner Gesprächsgruppen mit den Studenten erweitert, um die Erwartungshaltung und Anforderungen der Studenten zu ermitteln. Die Bedeutung der einzelnen Anforderungen für die Studenten wurde schließlich mithilfe eines Online-Fragebogens geklärt.

Der summative User-Test stellte einen Vergleich zwischen KonSearch und einem bestehenden System auf. Die Probanden sollten auf beiden Systemen typische Aufgaben durchführen, während ihre Aktionen aufgezeichnet wurden. Kombiniert wurde dies mit der Methode des Thinking Aloud. Ziel war, mittels der quantitativen Daten eine Aussage zu treffen, welches der beiden Systeme eine höhere Usability aufweist.

Ein ähnliches Vorgehen wies der formative Test auf. Auch hier wurde eine Protokollierung der Nutzeraktionen mit Thinking Aloud kombiniert. Allerdings wurde kein Vergleich durchgeführt, sondern einzig KonSearch betrachtet. Für eine Schlussfolgerung kamen dann qualitative Daten zur Anwendung.

Eine letzte Arbeitsgruppe setze für die Evaluation Eye-Tracking ein, um eine Bewertung des Design der Webseite vorzunehmen. So wurde zum Beispiel die optische Wirkung der Trefferlisten von KonSearch mit denen eines bestehenden Systems verglichen. Auch hier wurden den Probanden mehrere Aufgaben gestellt.

Abschließend erfolgte dann eine Zusammenfassung aller Ergebnisse der einzelnen Arbeitsgruppen. Die einzelnen Punkte wurden gewichtet und kategorisiert, sodass sich ein Gesamtbild der wichtigsten Probleme abzeichnete.

Kapitel 3

Verfahren der Zentralbibliothek

Nachdem im vorigen Kapitel die Thematik der Usability-Evaluationen näher beleuchtet wurde, soll sich dieses Kapitel mit dem Verfahren, wie es in der Zentralbibliothek zum Einsatz kommen soll, beschäftigen. Da das Informationsportal nur im Forschungszentrum eingesetzt wird, stellt dieses Kapitel als erstes JuLib eXtended näher dar, um eine Vorstellung zu liefern, für welche Webanwendung die Evaluation durchgeführt werden soll.

Da ein empirisches Verfahren zur Anwendung kommt, musste natürlich ein eigenes Verfahren entwickelt werden, welches für die Bedürfnisse der Zentralbibliothek ausgelegt ist. Dazu werden zunächst die bisher verfügbaren Möglichkeiten für Feedback aufgeführt und danach das zukünftige Verfahren mitsamt Details beschrieben.

3.1 JuLib eXtended

Das Informationsportal JuLib eXtended[1], im Folgenden nur noch kurz JuLib genannt, ist ein Angebot der Zentralbibliothek des Forschungszentrums Jülich. Technisch betrachtet wird die Homepage mithilfe des Frontends VuFind[2] ausgeliefert, während im Hintergrund die Datenhaltung von Apache Solr[3] vorgenommen wird. JuLib bietet den Mitarbeitern eine Möglichkeit nach Literatur zu suchen. Hier kann nicht nur der lokale Bestand durchsucht werden, sondern auch diverse externe Quellen. Zusätzlich kann der Nutzer im Katalog stöbern, falls er kei-

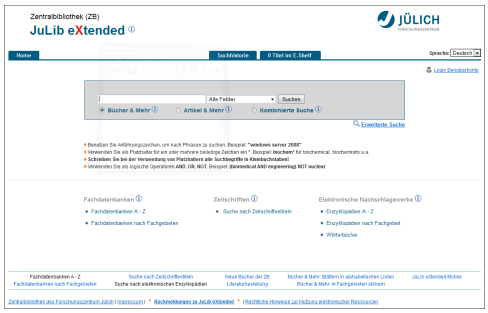

Abbildung 3.1: JuLib

[1] http://julib.fz-juelich.de/
[2] http://vufind.org/
[3] https://lucene.apache.org/solr/

ne genauen Suchangaben machen kann. Der Katalog umfasst dabei nicht nur Nachweise über physische Medien, sondern auch beispielsweise E-Books, bei denen der Volltext direkt eingesehen werden kann.

Neben diesen Möglichkeiten bietet JuLib auch personalisierte Funktionen. So können nach einer Anmeldung Medien sofort bestellt beziehungsweise vorgemerkt oder gegebenenfalls verlängert werden.

Genutzt wird JuLib überwiegend von den Mitarbeitern des Forschungszentrum, sowie einigen externen Personen. Die vorrangige Zielgruppe von JuLib sind die Wissenschaftler, aber auch Doktoranden, Studenten und Auszubildende, da sie am meisten mit Literatur in Kontakt kommen. Neben Deutsch ist auch Englisch in dieser Zielgruppe verbreitet.

3.2 Verfahren

Bevor das zukünftige Verfahren beschrieben wird, soll nochmals das bisherige Vorgehen mit Rückmeldungen erläutert werden. Auf JuLib gibt es in der Fußzeile, die auf jeder Seite angezeigt wird, die Möglichkeit, der Zentralbibliothek eine Rückmeldung zukommen zu lassen. Dahinter verbirgt sich kein gesondertes System, sondern einfache E-Mails. Diese laufen zentral an einer Stelle zusammen, werden dort einzeln verarbeitet und an die entsprechenden Stellen weitergeleitet. Zudem werden die Teilnehmer von Schulungen, die die Systeme der Zentralbibliothek betreffen, um eine Rückmeldung gebeten. Mit einem solchen passiven Feedback-System ist es allerdings nicht möglich, JuLib gezielt zu verbessern, da Nutzer eine Rückmeldung nur bei erheblichen Mängeln abgeben. Daher ist ein Verfahren, welches aktiv auf die Nutzer zugeht, besser zur Ermittlung der Usability geeignet.

Das in dieser Arbeit eingesetzte Verfahren wurde speziell für die Zentralbibliothek entworfen. Als Erstes musste dafür die Art der Usability-Evaluation festgelegt werden. Das hier vorgestellte Verfahren ist empirischer Natur. Da keine finanziellen Mittel für geschulte Usability-Experten verfügbar sind, und dies auch kein internes Personal übernehmen kann, scheidet ein analytisches Verfahren aufgrund expliziter Expertentester aus.

Als Probanden sollen Besucher des Lesesaals der Zentralbibliothek hinzugezogen werden, die sich aus allen am Campus angesiedelten Fachbereichen zusammensetzen. Somit ist die Zielgruppe für den Test nicht homogen, obwohl festgestellt werden kann, dass der Großteil aus einem wissenschaftlichen Umfeld mit akademischem Abschluss stammt.

Die Besucher könnten beispielsweise von den Lesesaal-Mitarbeitern angesprochen werden, ob sie bei einem solchen Test mitmachen würden. Da die Zentralbibliothek sehr viele Schulungen und Lehrgänge anbietet, können auch dort Probanden gewonnen wer-

den. Wenn direkt nach der Schulung die Teilnehmer einen Test bearbeiten, könnten daraus auch Rückschlüsse auf den Erfolg der Schulung gezogen werden. Im Gegensatz zu den Besuchern des Lesesaals, bilden die Teilnehmer einer Schulung unter bestimmten Kriterien eine homogene Menge, was bei einem Test berücksichtigt werden kann.

Neben der Einordnung zu den empirischen Verfahren kann die Methode der Zentralbibliothek noch zu den summativen Verfahren gezählt werden. Zum einen befindet sich JuLib nicht mehr in der Entwicklung, sondern bereits im produktiven Einsatz, und zum anderen kann von den Besuchern keine qualitative Meinung verlangt werden. Stattdessen wird das Verfahren so aufgebaut, dass quantitative Daten ermittelt werden, welche auf einfache Weise erfasst werden können, ohne dass die Probanden diese selbst liefern müssen.

Für die eigentliche Konzeption des Verfahrens mussten die verfügbaren Mittel und Gegebenheiten betrachtet werden. Hierbei wurde festgestellt, dass in der Zentralbibliothek keine dedizierten Räumlichkeiten für einen Test mit Probanden unter Laborbedingungen zur Verfügung gestellt werden können. Auch soll eine spätere Auswertung mit möglichst wenig Aufwand durchführbar sein, indem diese automatisch geschieht und ohne manuelle Arbeit auskommt. Daher musste ein Verfahren entworfen werden, welches dies nicht erfordert und auf Techniken, wie beispielsweise Eye-Tracking oder Aufnahmen, verzichtet. Wie bereits erwähnt, sollen als Probanden Besucher des Lesesaals oder Teilnehmer einer Schulung gewonnen werden, von daher liegt es nahe, direkt einen im Lesesaal zur Verfügung stehenden PC oder Laptop zu nutzen.

Bisher: passiv	**Zukünftig: aktiv**
Feedback-Formular	Analyse der Aufgabenbearbeitung
Direktes Feedback	dynamische Formulare

Tabelle 3.1: Evaluations-Methoden

So wurde ein Test konzipiert, welcher von den Probanden selbstständig durchgeführt werden kann und bei der Protokollierung nicht von der Umgebung abhängt. Dazu wird der Kern des Verfahrens von Aufgaben gebildet. Die Aufgaben sollen von den Probanden bearbeitet werden, währenddessen alle Aktionen dieser im Hintergrund protokolliert werden. Zusätzlich zu den Aufgaben ist es möglich, Formulare zu integrieren, die von den Teilnehmern ausgefüllt werden sollen. Ein Formular soll zu Beginn des Tests ein paar persönliche, aber anonyme Daten erfassen. Ein anderes Formular soll am Ende ein Feedback abfragen. Zudem sind weitere Feedback-Formulare nach jeder Aufgabe denkbar. Dies ist ein besonders wichtiger Aspekt, da manche Gegebenheiten nicht aus den objektiven und distanzierten Messwerten ersichtlich sind, sondern von den rein subjektiven Gefühlen der

Testpersonen abhängen. Nachdem eine Testperson alle Aufgaben beendet hat, sind keine weiteren Schritte wie etwa ein persönliches Gespräch geplant, und die Person ist aus dem Test entlassen.

Alle weiteren Schritte zur Auswertung sowie ein Severity-Rating finden dann unabhängig im Hintergrund statt. Sie werden von dieser Arbeit nicht näher erläutert.

Die Evaluations-Methoden, die bis jetzt eingesetzt werden und die zum Einsatz kommen sollen, sind in der Tabelle 3.1 nochmals zusammengefasst.

3.3 Aufgaben

Die Aufgaben bilden den Schwerpunkt bei der hier verwendeten Usability-Evaluation. Im Allgemeinen beschreiben die Aufgaben Tätigkeiten, die auf der entsprechenden Webseite durchgeführt werden sollen. Sie bieten dem Probanden einen Leitfaden für die Aktionen, die er unternimmt. Zudem unterstützen sie die spätere Auswertung in hohem Maße. Als Alternative würde sich eine freie Beobachtung zufälliger Lesesaal-Besucher anbieten, nur würde dort der feste Rahmen fehlen. Somit wäre bei der späteren Auswertung der objektiven Messwerte nicht nachvollziehbar, was der Nutzer eigentlich machen wollte. Außerdem wäre eine Bewertung zwischen Zielsetzung und wirklichen Handlungen nicht mehr möglich. Aus diesem Grund werden hier Aufgaben eingesetzt, die einen idealen Handlungspfad durch die Weboberfläche besitzen, mit dem dann die aufgezeichneten Aktionen verglichen werden können. Damit für den Fall, dass eine Aufgabe vom Teilnehmer nicht abgeschlossen werden kann, der Test trotzdem weiterläuft, gibt es die Möglichkeit, eine Aufgabe gegebenenfalls zu überspringen.

Das Erstellen von Aufgaben ist keine Trivialität [Heg03, 39 f.]. Generell sollen die Aufgaben genügend Informationen besitzen, um dem Tester diese zu verdeutlichen, ohne den Weg gänzlich vorzuschreiben. In der Praxis muss zwischen diesen beiden Aspekten abgewogen werden. Erfolgt die Formulierung der Aufgaben sehr präzise, ist die Auswertung einfacher zu generalisieren, aber dadurch wird der Test zu realitätsfern, da der Proband zu sehr in seinen Handlungen eingeschränkt wird. Sind die Aufgaben auf der anderen Seite sehr offen gefasst, muss die Testperson selbstständig das Ziel erreichen, allerdings geht so die Vergleichbarkeit verloren, da beispielsweise mehrere Wege oder Lösungen der Aufgabenstellung genügen könnten. Neben der reinen Aufgabenstellung kann der Text einer Aufgabe auch einen Kontext bereitstellen, um dem Probanden einen gewissen Hintergrund zu liefern. Ein Beispiel dazu wäre folgende Aufgabe:

> Ich sitze an meinem Arbeitsplatz und interessiere mich für das Buch „*Aus weniger mehr machen: Strategien für eine nachhaltige Ressourcenpolitik in*

Deutschland“ von Peter Hennicke und möchte es ausleihen.
Bestellen Sie das Buch bei der Zentralbibliothek.

Statt einfach nur die Aufgabe „leihen Sie das Buch [..] aus“ zu stellen, bekommt der Proband noch ein wenig Hintergrund geliefert. So wird ihm die Aufgabestellung erklärt, und er kann sich besser hineinversetzen.

3.4 Messdaten

Um eine möglichst genaue Auswertung der Tests und damit gute Ergebnisse der Usability-Evaluation zu ermöglichen, werden vielseitige Messdaten erhoben. Alle Messdaten stammen dabei aus der Protokollierung der Aktionen des Probanden. Als Hauptbestandteil ist allen Messdaten ein zugeordnetes Ereignis gemein, und zwar bei welchem Ereignis der Datensatz anfiel. Grundsätzlich lassen sich alle Ereignisse in zwei Kategorien einteilen: Zum einen in Ereignisse, die bei der Bedienung der Evaluations-Umgebung anfallen und zum anderen Ereignisse, die auf der zu testenden Webseite ausgelöst werden.

Ereignis	**Bedeutung**
unitLoaded	Testabschnitt geladen
formFinished	Formular abgeschickt
taskStarted	Aufgabe gestartet
taskFinished	Aufgabe abgeschlossen
taskSkipped	Aufgabe übersprungen
click	Mausklick
keyDown	Tastendruck
nav	Browser-Navigation

Tabelle 3.2: Ereignisse

Zu der ersten Kategorie gehören die Ereignisse *unitLoaded*, *formFinished*, *taskStarted*, *taskFinished* und *taskSkipped*. Das Ereignis *unitLoaded* wird immer dann ausgelöst, wenn ein neuer Testabschnitt beginnt, also eine neue Aufgabe oder neues Formular geladen wird. Wohingegen die drei Ereignisse *taskStarted*, *taskFinished*, *taskSkipped* angeben, ob eine Aufgabe gestartet, erfolgreich beendet oder übersprungen wurde. Als letztes Ereignis dieser Kategorie zeigt *formFinished* das Abschicken eines Formulars an.

In der zweiten Kategorie befinden sich die Ereignisse *nav*, *click* und *keyDown*. Die Ereignisse *click* und *keyDown* treten auf, sobald die Testperson mit der Maus klickt beziehungsweise etwas tippt. Das Ereignis *nav* protokolliert die Navigation über die verschiedenen Unterseiten der zu testenden Webseite, wird also immer dann ausgelöst, wenn die Seite geändert wird.

Zu allen Ereignissen werden neben der Art noch weitere Metadaten gespeichert. Dies sind zum einen die URL, auf der sich der Proband momentan befindet, sowie ein Zeit-

stempel. Auch ein Text mit weiteren Informationen wird zu jedem Ereignis hinterlegt, dessen Inhalt vom Typ des Ereignisses abhängt.

Neben der Protokollierung aller Aktionen bei der Bearbeitung der Aufgaben werden alle Daten, die beim Ausfüllen von Formularen anfallen, gesichert. Um den Datenschutz zu gewährleisten, werden alle Messdaten anonym gespeichert und sensible Daten wie Anmeldeinformationen nicht protokolliert.

Kapitel 4

Implementierung – Grundlagen

Bevor in den nächsten Kapiteln näher auf die praktische Realisierung des in dieser Arbeit angewendeten Verfahrens eingegangen wird, beschäftigt sich dieses Kapitel zunächst mit den Grundlagen, die für eine erfolgreiche Implementierung notwendig sind. Zunächst werden alle verwendeten Technologien wie Programmiersprachen oder Kommunikationsmethoden erklärt. Dies ergänzend wird dann die Entwicklungsumgebung, die für die Umsetzung zur Verfügung steht, beschrieben. Nachdem die Frage „Womit?“ geklärt ist, wird auf das „Wie?“ eingegangen. In diesem Zuge wird als erstes das zugrundeliegende Architekturmuster vorgestellt und danach die Konzeption für die Implementierung, wie sie schließlich gewählt wird, aufgegriffen.

4.1 Verwendete Technologien

HTML

Die Hypertext Markup Language[1] (HTML) ist eine textbasierte Auszeichnungssprache für elektronische Dokumente. HTML ist seit 1995 standardisiert und ist aktuell in Version 4.01 (Stand Juli 2014) verfügbar. Version 5 befindet sich in Entwicklung und wird in den gängigen Programmen bereits teilweise unterstützt. HTML dient als Grundlage des World Wide Web und wird üblicherweise vom Webbrowser dargestellt. Da reines HTML nur statische Inhalte bietet, wird sie meist mit Zusatztechniken wie JavaScript erweitert. Eine Übergabe von Werten an den Server ist in HTML mit zwei Methoden möglich. Die Methode POST überträgt die Daten im Hauptteil der Anfrage, während bei GET die Daten bereits in der URL übergeben werden.

[1] http://de.wikipedia.org/wiki/Hypertext_Markup_Language

DOM

Das Document Object Model[2] (DOM) ist eine Norm für den Zugriff auf beliebige Elemente eines HTML- beziehungsweise XML-Dokuments. Diese Schnittstelle definiert lediglich Objekte, Eigenschaften sowie Methoden, die für eine Implementierung notwendig sind. Alle Elemente sind dabei ihrer Verschachtelung folgend als Baum organisiert, über welchen traversiert werden kann. Aber auch ein freier Zugriff auf beliebige Elemente ist möglich.

JavaScript

JavaScript[3] ist eine clientseitige Skriptsprache. Sie wird direkt mit in das HTML-Dokument integriert und dient zur Erstellung dynamischer Webseiten. Mit JavaScript können zur Laufzeit beliebige Änderungen an dem DOM von HTML-Dokumenten vorgenommen werden, indem Eigenschaften von Elementen geändert werden oder auf Ereignisse, wie Mausklicks oder Tastendrücke, reagiert wird. Wichtig ist, dass JavaScript clientseitig läuft, und somit auch vom Webbrowser interpretiert wird. Um die Sicherheit des Anwenders zu erhöhen, wird der Quellcode in einer Sandbox – jede Webseite läuft isoliert – ausgeführt, was aber auch zu Einschränkungen des Funktionsumfangs führt. So können dynamische JavaScript-Inhalte nur auf die eigenen Webseiten angewendet werden, aber nicht auf eingebundene externe Inhalte[4].

AJAX

AJAX[5] steht für „Asynchronous JavaScript and XML“ und beschreibt ein Konzept zur asynchronen Datenübertragung zwischen Webbrowser und Server. Bei normaler synchroner Nutzung des Browsers wird eine Webseite nach der anderen im Vordergrund geladen. Bei AJAX hingegen führt eine JavaScript-Routine im Hintergrund weitere Anfragen aus (Abb. 4.1), ohne dass der Nutzer dies direkt mitbekommt. So können für die Bequemlichkeit der Nutzung einzelne Teile der Webseite aktualisiert werden und die Seite muss nicht komplett neu geladen werden.

[2] http://de.wikipedia.org/wiki/Document_Object_Model
[3] http://de.wikipedia.org/wiki/JavaScript
[4] http://de.wikipedia.org/wiki/Cross-Site-Scripting
[5] http://de.wikipedia.org/wiki/Ajax_%28Programmierung%29

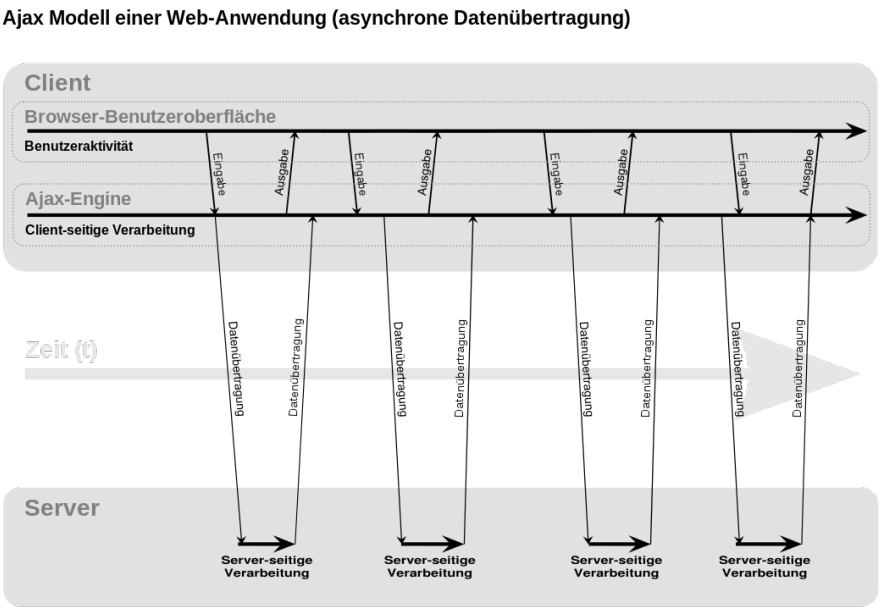

Abbildung 4.1: Prozessfluss AJAX[Dan07]

Prototype-Framework

Das JavaScript-Framework Prototype[6] stellt viele Methoden zur Vereinfachung von JavaScript-Quellcode zur Verfügung, indem es komplexere Konstrukte direkt bereitstellt. Darüber hinaus bietet es auch vorgefertigte Routinen für den Einsatz von AJAX. Es ist aktuell in Version 1.7.2 (Stand Juli 2014) verfügbar.

CSS

Cascading Style Sheets[7] (CSS) helfen bei der Trennung von Aussehen und formaler Beschreibung von Webseiten. CSS ist eine deklarative Sprache zur Erstellung von Stilvorlagen für strukturierte Dokumente, hauptsächlich HTML-Dokumente. Mit CSS können für bestimmte Elemente des HTML-Dokuments Design-Regeln aufgestellt werden, ohne dass diese direkt in HTML festgelegt werden müssen. So kann für eine komplette Homepage das Aussehen allein durch den Austausch der CSS-Vorlagen geändert werden. Außerdem können in CSS verschiedene Ausgabemedien beachtet werden, sodass sich das Design dem Medium anpasst.

[6] http://prototypejs.org/
[7] http://de.wikipedia.org/wiki/Cascading_Style_Sheets

PHP

PHP[8](„PHP: Hypertext Preprocessor“) ist, wie der Name schon sagt, eine Vorstufe zu HTML. Es dient zur Erzeugung dynamischer Webseiten, indem PHP-Quellcode in HTML-Dokumente eingefügt und zur Laufzeit interpretiert wird. PHP ist eine serverseitige Skriptsprache und die aktuelle Version ist 5.5.13 (Stand Juli 2014). Ursprünglich 1995 von Rasmus Lerdorf als Ersatz für eine Sammlung an Perl-Skripten geschaffen, ist PHP mittlerweile eine eigenständige und weltweit die meist verbreitete serverseitige Programmiersprache für Webseiten[9]. Sie bietet neben dem sehr großen Funktionsumfang auch viele Möglichkeiten zur Datenbankanbindung, eine hohe Stabilität und kommt deswegen auch bei dieser Arbeit zum Einsatz.

Smarty

Smarty[10] ist eine Template-Engine für PHP. Sie dient zur Umsetzung des MVC-Musters (Model-View-Controller[11]), das heißt zur Trennung von Programmlogik und Präsentation. Dies geschieht, indem nun nicht mehr PHP direkt in die HTML-Dokumente eingefügt wird, sondern Smarty. Es ist speziell zur Erzeugung von Webseiten ausgelegt und verfügt auch über einfache Konstrukte, wie if-Abfragen oder Schleifen. Der Smarty-Quellcode wird zur Laufzeit in ein PHP-Dokument kompiliert, und danach mit dem PHP-Interpreter weiterverarbeitet. Der PHP-Quellcode steht in einer separaten Datei und übergibt dem Smarty-Template alle Daten die angezeigt werden sollen. Somit können beliebige Teile wie zum Beispiel die komplette Logik ausgetauscht werden, ohne die anderen Teile zu beeinträchtigen. Es kommt Smarty in Version 3.1.8 zum Einsatz.

SQL

SQL[12] wird bei relationalen Datenbanken als Datenbanksprache genutzt. Mit ihr können Datenstrukturen definiert, bearbeitet sowie abgefragt werden. Als Basis dient die relationale Algebra, ihr Aufbau ist aber einfach und an die englische Umgangssprache angelehnt. Obwohl SQL standardisiert ist, hängt die wirkliche Ausgestaltung stark vom verwendeten Datenbanksystem ab. In dieser Arbeit kommt der SQL-Dialekt von MySQL zum Einsatz.

[8] http://php.net
[9] http://w3techs.com/technologies/overview/programming_language/all
[10] http://smarty.net
[11] http://de.wikipedia.org/wiki/Model_View_Controller
[12] http://de.wikipedia.org/wiki/SQL

XML

Die Extensible Markup Language[13] (XML) ist eine Auszeichnungssprache ähnlich zu HTML. Sie dient zur Notation von hierarchisch strukturierten Daten. Vor allem im Internet wird XML eingesetzt, um einen plattformübergreifenden Datenaustausch mittels einfachem Text zu ermöglichen. Ergänzt wird XML durch die Schemasprache DTD (Dokumenttypdefinition), mit welcher ein allgemeines Schema einer XML-Datei festgelegt werden kann. So können beliebige XML-Daten gegen eine DTD validiert und eine Überprüfung, ob diese dem Schema genügen, durchgeführt werden. Neben der DTD gibt es auch die XSD[14] (XML Schema Definition). Dies ist eine weitere Möglichkeit XML-Dokumente zu definieren, die weit komplexer aufgebaut ist als DTD und so besser den Sachverhalt abbilden kann. So können einfache oder komplexe Datentypen erstellt werden, die den Elementen zugeordnet werden, aber auch Schlüssel und Erweiterungen von bestehenden Schemata.

4.2 Entwicklungsumgebung

Das Programm wird vollständig unter Red Hat Enterprise Linux 6 64 Bit entwickelt. Zum Einsatz kommt dabei Gedit 2.28.4 als Editor und Apache der Version 2.2.15 als Webserver. Auf dem Webserver wird PHP 5.5.12 in Verbindung mit MySQL in Version 5.5.37 verwendet. Getestet wird das Programm ausschließlich mit dem Firefox 31.0 (oder neuer), weshalb die korrekte Ausführung nur mit diesem Webbrowser garantiert wird. Für Testzwecke wurde eine komplette Spiegelung von JuLib auf dem Testsystem eingerichtet, mit welcher gearbeitet wird.

4.3 Architekturmuster

Um bei dem zu entwickelnden System eine saubere Gliederung und damit eine gute Umsetzung zu gewährleisten, wird für das Programm ein Architekturmuster als Grundlage genommen. Architekturmuster[15] beschreiben kein konkretes Vorgehen, sondern die zugrundeliegende Organisation der einzelnen Komponenten und wie diese

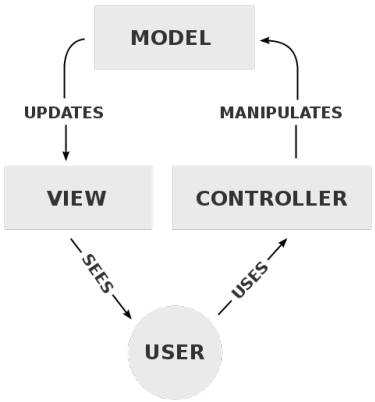

Abbildung 4.2: MVC-Model[Reg10]

[13] http://de.wikipedia.org/wiki/Extensible_Markup_Language
[14] http://de.wikipedia.org/wiki/XML_Schema
[15] http://de.wikipedia.org/wiki/Architekturmuster

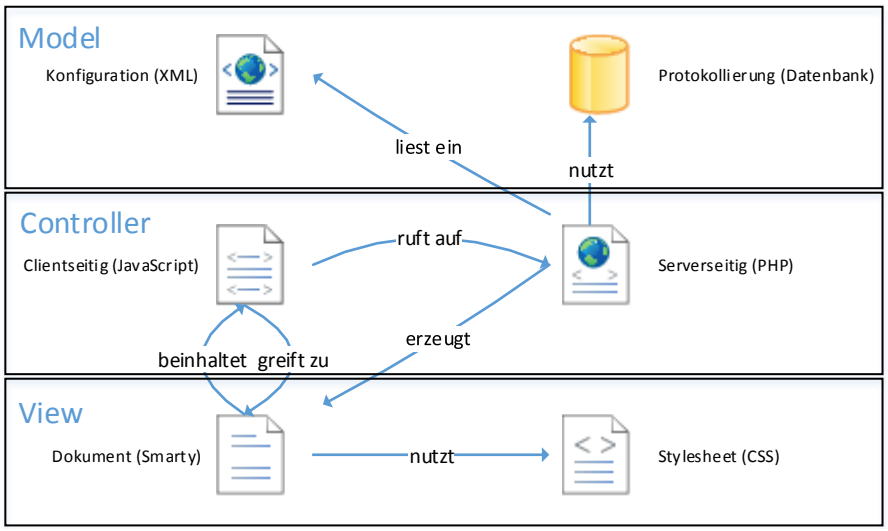

Abbildung 4.3: Architektur der Testumgebung

zusammenhängen.

Als Architekturmuster wird für diese Arbeit das MVC-Model (Model-View-Controller) gewählt. Wie der Name schon sagt, besteht dieses aus den drei Bestandteilen Model, View und Controller. Eine Darstellung, wie die einzelnen Teile zusammenwirken, ist in Abbildung 4.2 zu sehen.

Das Model umfasst die eigentlichen Daten und ist vollständig unabhängig vom Controller und der View. Es wird hier mithilfe einer MySQL-Datenbank abgebildet, welche alle anfallenden Daten speichert. Dabei wird die Konfiguration der Tests per XML-Datei vorgenommen. Dargestellt werden die Daten in der View. Im Regelfall kennt die View sowohl den Controller als auch das Model. Die View nimmt auch Nutzereingaben entgegen und gibt diese ohne Verarbeitung an den Controller weiter. In dieser Arbeit besteht die View aus Smarty und CSS-Vorlagen, die zusammen als HTML-Dokument dargestellt werden. Als zentrales Element schließlich verwaltet der Controller die gesamte Logik. Er verarbeitet alle Daten, speichert sie im Model und zeigt diese per View an. Umgesetzt wird der Controller auf der Serverseite mit PHP-Skripten. Clientseitig wird der Controller mittels JavaScript realisiert, dort ist er vor allem für die Protokollierung zuständig. Der komplette Aufbau des Architekturmusters ist in Abbildung 4.3 dargestellt.

4.4 Konzeption der Implementierung

Nachdem in dem vorigen Kapitel 3 das Verfahren der Zentralbibliothek näher beschrieben und in diesem Kapitel bereits auf einige technische Grundlagen eingegangen wurde, soll sich dieser Abschnitt der Frage der Konzeption der Implementierung widmen. Dabei soll nicht die genaue Realisierung im Mittelpunkt stehen, sondern vielmehr die Gedanken, die vorweg getätigt wurden.

Als eine ungefähre Vorstellung vorhanden war, wie die Usability-Evaluation ablaufen soll, wurde zunächst ein Blick auf bestehende Tools[16] für diesen Zweck geworfen. Es gibt viele, teils kommerzielle, Programme zu diesem Thema. Ein Teil der Programme dient zur Durchführung von einfachen Umfragen, aber viele haben auch interessantere Ansätze. So sind durchaus auch Programme zu finden, die Prognosen für die Nutzerbewegung über eine Webseite geben sollen oder sogenannte „Heatmap“ (siehe Abb. 4.4) zu den Mausklicks von Nutzern erstellen.

Abbildung 4.4: Beispiel einer Heatmap [Pen06]

Für die Zwecke dieser Arbeit sollen weder kommerzielle noch freie Programme zum Einsatz kommen. Während kommerzielle bereits durch ihre Kosten ausscheiden, ist es bei den freien Programmen vor allem eine Frage der Wartbarkeit. Bei freien Programmen ist es oftmals so, dass für sie kein Support verfügbar ist, und etwaige Fehler und Probleme nur mit erheblichem Aufwand behoben werden können, wenn dies überhaupt möglich ist. Statt dessen soll ein eigenes Tool entwickelt werden, welches speziell auf die in dieser Arbeit entwickelte Methode angepasst ist.

Es gibt eine Reihe Tools, die eine Konfiguration bzw. Installation auf dem lokalen System des Probanden erfordern. Eine solche clientseitige Veränderung des lokalen Systems ist aber in diesem Fall nicht erwünscht. Dadurch kommt nur eine serverseitige Lösung in Frage.

Betrachtet man die Funktionsweise von vielen serverseitigen Tools, so müssen diese meistens auf dem eigenen Webserver laufen, während diese die zu testende Webseite mit JavaScript-Code erweitern. Dieses Vorgehen soll auch für das hier zu entwickelnde Tool benutzt werden. Wie bereits erläutert, sollen alle Aktionen, die die Probanden ausführen, protokolliert werden, und die Möglichkeit gegeben sein, Formulare einzubauen. Ersteres könnte per Proxy realisiert werden. Der Proxy sitzt zwischen dem lokalen PC des Proban-

[16] Eine Liste mit einigen Tools ist hier zu finden: http://www.hongkiat.com/blog/tools-to-improve-your-websites-usability/

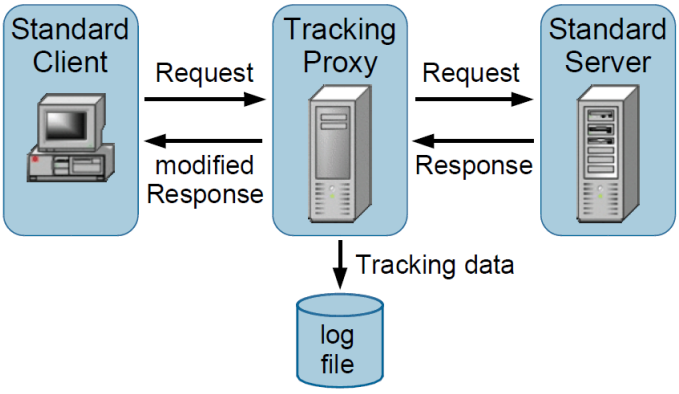

Abbildung 4.5: Skizze des Proxy [AWS06]

den und dem Server, wobei der komplette Netzwerkverkehr zwischen den beiden Enden über ihn läuft. Auf diese Weise hat der Proxy die Möglichkeit alle HTML-Dokumente die der lokale PC vom Server empfängt zu modifizieren und mit JavaScript-Quellcode zur Überwachung anzureichern (vgl. Abb. 4.5). Allerdings ist es mit dieser Methode nicht möglich, eigene Formulare an die Probanden zu reichen. Für dies hätte neben dem Proxy eine zusätzliche Methode genutzt werden müssen. Aus diesem Grund wurde schließlich keine Proxy-Lösung umgesetzt. Stattdessen wird eine andere Variante gewählt, die alle Anforderung erfüllt: keine clientseitigen Veränderungen, dynamische Formulare, Protokollierung der Ereignisse sowie keine Erhebung von personenbezogener Daten.

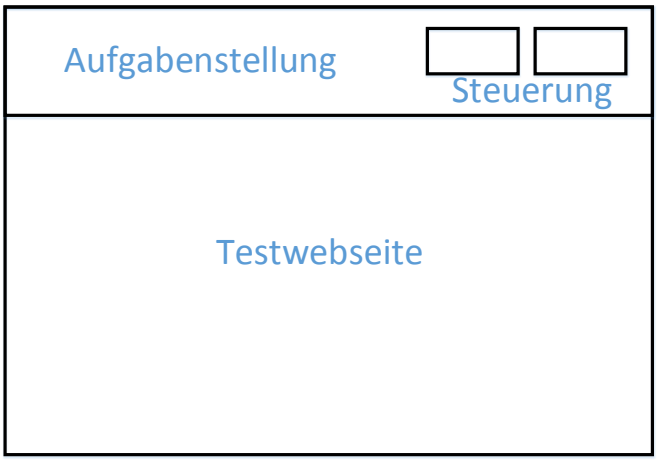

Abbildung 4.6: Skizze der Aufgaben-Oberfläche

Die Evaluations-Umgebung wird als selbstständige Webseite entworfen. Somit können Formulare direkt auf dieser angezeigt werden, während die zu testende Webseite per IFra-

me eingebunden wird. Zusätzlich kann dann während eines laufenden Tests der Text der aktuellen Aufgabe am oberen Rand angezeigt werden (vgl. Abb. 4.6). Die Protokollierung erfolgt nun per JavaScript, welches das IFrame instrumentalisiert und so alle Aktionen protokolliert. Diese Art der Umsetzung beherbergt allerdings eine Unbequemlichkeit, die durch den Einsatz eines IFrame zur Darstellung der zu testenden Webseite bedingt wird. Durch die Same-Origin-Policy[17] von JavaScript muss die Evaluations-Umgebung auf dem selben Server liegen wie die zu testende Webseite, da sonst keine Protokollierung von Nutzeraktionen erfolgen kann.

Um einen eindeutigen Rahmen um die Bearbeitung einer Aufgabe zu spannen, muss festgelegt werden, an welcher Stelle eine Aufgabe beginnt und wo sie endet. Es muss also der Start- und Zielpunkt jeder Aufgabe wohldefiniert sein. Als ein einfaches Verfahren bietet sich die Definition über die URL an. Der Startpunkt wird mittels einer eindeutigen URL angegeben, während die Zieladresse als regulärer Ausdruck festzulegen ist. Ein regulärer Ausdruck an dieser Stelle ermöglicht mehrere Enden der Aufgabe und eine Unabhängigkeit von dynamischen URL-Inhalten.

[17] https://de.wikipedia.org/wiki/Same-Origin-Policy

Kapitel 5

Implementierung – Datenhaltung

Die Datenhaltung spielt bei einer Usability-Evaluation eine entscheidende Rolle. Ohne eine gute Datenhaltung sind alle Messdaten, die während der Evaluation erhoben werden, wertlos, da aus ihnen dann keine aussagekräftigen Rückschlüsse gezogen werden können. Somit sollte vorab bereits ein gutes Konzept zur Verwaltung der Daten erarbeitet werden. Bei dieser Arbeit unterteilt sich die Datenhaltung in zwei Bereiche. Der erste Bereich umfasst die Konfiguration der Evaluation, also Daten, die bereits im Vorfeld des eigentlichen Tests festgelegt werden. Der zweite Bereich hingegen beinhaltet alle Daten, die während des Tests erfasst werden. Dies sind zum einen sämtliche Messdaten, die erhoben werden, aber auch Formularinhalte sowie Hilfsdaten, die für die Organisation wichtig sind.

In den folgenden Abschnitten wird zunächst auf die Konfiguration und danach auf die Datenhaltung eingegangen.

5.1 Konfiguration – XML

Statt die einzelnen Tests fest vorzugeben, soll eine Konfiguration ermöglicht werden, wobei diese ohne Hintergrundwissen und Hilfsmittel möglich sein soll. Daher wurde von einer Konfiguration über eine Datenbank abgesehen. Zum einen müsste dafür ein umfangreiches Datenbankschema entworfen werden, zum anderen hätte noch eine dazu passende Oberfläche erzeugt werden müssen, da sonst das Eintragen von Daten nur mit hohem Aufwand möglich ist. Deshalb wird die Konfiguration der Testumgebung von XML-Dateien vorgenommen. XML bietet den Vorteil, dass einzelne Teile der Konfiguration schnell in dem strukturierten Format angepasst werden können, aber kein Fachwissen dazu vonnöten ist.

Es wird eine XML-Datei pro Testsuite[1] benötigt. Eine Testsuite umfasst dabei einen kompletten Testablauf. Dieser besteht aus einer linearen Anordnung von beliebig vielen

[1] http://en.wikipedia.org/wiki/Test_suite

Aufgaben und Formularen. Das genaue Format der XML-Datei ist in einer Dokumenttypdefinition festgelegt, welche in Anhang B hinterlegt ist. Der Aufbau gliedert sich in folgende Elemente:

5.1.1 XML-Element Testsuite

Hauptelement der XML-Datei. Es besitzt die beiden optionalen Attribute *description* und *blacklist*. *description* gibt eine einfache Beschreibung der Testsuite an, welche auch später in der Datenbank übernommen wird. *blacklist* beinhaltet eine Liste an Webseiten-Elementen, welche nicht mitprotokolliert werden sollen. Dies könnten zum Beispiel die Eingabefelder eines Logins sein. Die Elemente müssen entweder per HTML-Attribut *name* oder *id* angegeben werden. Einzelne Listenelemente werden per Leerzeichen voneinander getrennt. Als Unterelemente sind die Elemente *form* und *task*, samt Wiederholungen dieser, zulässig. Die Reihenfolge der Unterelemente stellt auch gleichzeitig die Reihenfolge des Testes dar: Der Proband bekommt also die einzelnen Bestandteile in genau dieser Reihenfolge präsentiert.

```
<testsuite description="<!-- Beschreibung -->" blacklist="<!--
    Elemente -->">
  <!-- form- oder task-Elemente -->
</testsuite>
```

5.1.2 XML-Element Form

Um den Erstellern von Testsuites bei der Umsetzung ihrer Ideen entgegenzukommen, gibt es keine vorgefertigten Formulare. Stattdessen können diese komplett dynamisch erstellt werden, so können den Probanden immer situationsgerechte Formulare präsentiert werden. Ein solches Formular wird mit dem *form*-Element erstellt. Als Pflichtattribut benötigt dies das Attribut *name*, unter welchem die Formulardaten in der Datenbank gespeichert werden. Des Weiteren kann mit dem Attribut *prompt* eine Abfrage vor das Formular geschaltet werden, mit dem das Formular übersprungen werden kann. Der Wert des Attributs ist dabei der Text des Dialogs.

Unterhalb dieses Elements gibt es die beiden Elemente *input* und *text*. Per *text*-Element kann ein einfacher statischer Text angezeigt werden. Diesem kann noch eine optionale Überschrift gegeben werden, indem das Attribut *label* verwendet wird. *input* hingegen wird für jedwede Eingabemöglichkeit seitens der Nutzer benutzt.

Als Pflichtattribut muss *type* angegeben werden. *type* legt fest, von welcher Art die

Eingabemöglichkeit ist. Erlaubt sind die Werte „text“, „checkbox“, „radio“, „select“ und „textarea“. Was sich genau hinter diesen Angaben verbirgt zeigt die Abbildung 5.1.

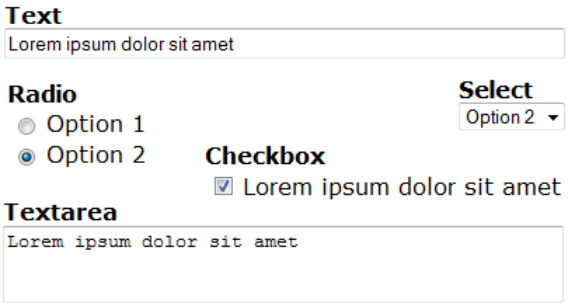

Abbildung 5.1: Input-Typen

Des Weiteren kann mit dem Attribut *name* der Name angegeben werden, unter welchem die Protokollierung in der Datenbank erfolgt. Wird *name* nicht gegeben, wird stattdessen der Wert des Attributs *label* (gleiche Funktion wie bei dem Element *text*) genommen. Zu beachten ist, dass keine Protokollierung dieses Elements stattfindet, wenn keins dieser beiden gesetzt sein sollte. Zuletzt ist bei dem Typ „checkbox“ das Attribut *checked* möglich, mit den beiden Werten „true“ oder „false“. Dieses gibt an, ob das Feld standardmäßig angekreuzt sein soll oder nicht. Wird das Attribut nicht angegeben, ist das Verhalten gleich dem Zustand „false“. Bei dem Element *input* gibt der Inhalt den Standard-Inhalt der Eingabemaske an bzw. beim Typ „checkbox“ den Text hinter dem Kästchen. Nur die Typen „radio“ und „select“ bilden hierbei eine Ausnahme, da bei ihnen *option*-Elemente als Unterelemente angeben werden müssen, um die verschiedenen Auswahlmöglichkeiten zu definieren.

Insgesamt können beliebig viele Unterelemente in einem *form*-Element vorkommen, auch die Reihenfolge ist nicht vorgegeben.

```
<form name="<!-- Name -->">
  <text>Statistische Daten</text>
  <input type="text" label="Ihr Alter"/>
  <input type="select" label="Akademischer Grad">
    <option>Bachelor</option>
    <option>Master</option>
  </input>
  <submit>Ok</submit>
</form>
```

5.1.3 XML-Element Task

Dieses Element erzeugt eine Aufgabe, die der Proband lösen soll. Selber besitzt das Element *task* keine Attribute, sondern wird ausschließlich über seine Unterelemente näher beschrieben. Die Unterelemente müssen hierbei alle genau einmal vorkommen. Dies wären die Elemente *start*, *end* sowie *text*. Die Elemente *start* und *end* geben den Startpunkt bzw. den Endpunkt des Testes an, beinhalten also jeweils eine URL. Der Startpunkt wird dabei über eine eindeutige URL spezifiziert, während der Endpunkt über einen regulären Ausdruck angegeben wird. Allerdings muss beachtet werden, dass gegebenenfalls Zeichen den Regeln für reguläre Ausdrücke[2] entsprechend maskiert werden müssen. Zuletzt gibt das Element *text* noch den eigentlichen Text an, den der Proband angezeigt bekommt.

```
<task>
  <start>http://localhost/start.php</start>
  <end>http://localhost/end.php</end>
  <test><!-- Aufgabe --></text>
</task>
```

5.2 Protokollierung – Datenbank

In diesem Abschnitt wird näher auf die Datenhaltung eingegangen. Diese umfasst alle Messdaten, die in Kapitel 3.4 aufgeführt sind. Als Bedingung an die Datenhaltung wird eine einfache Zugriffsmethode gestellt, sodass alle erhobenen Daten für eine spätere Auswertung ohne Aufwand ausgelesen werden können. Dazu zählt auch ein partieller Zugriff auf die Daten, um gezielt einzelne Aspekte zu analysieren. Aus diesem Grund wird eine SQL-Datenbank zur Datenhaltung gewählt, welche mit MySQL realisiert wird. Als Speichersubsystem kommt InnoDB zum Einsatz, um Fremdschlüssel einsetzen zu können.

1. Normalform	Attribute haben atomaren Wertebereich.
2. Normalform	1. NF + Nichtschlüsselattribute sind von allen Schlüsselattributen voll funktional abhängig.
3. Normalform	2. NF + kein Nichtschlüsselattribut ist von einem Schlüsselattribut transitiv abhängig.

Tabelle 5.1: Normalformen

Die Tabellen, die die Datenbank enthält, wurden so entworfen, dass sie der dritten Normalform (siehe Tab. 5.1) genügen. Die Abbildung 5.2 zeigt das komplette Datenbankschema, welches nun im Einzelnen detailliert erläutert wird.

[2] http://www.regexr.com/

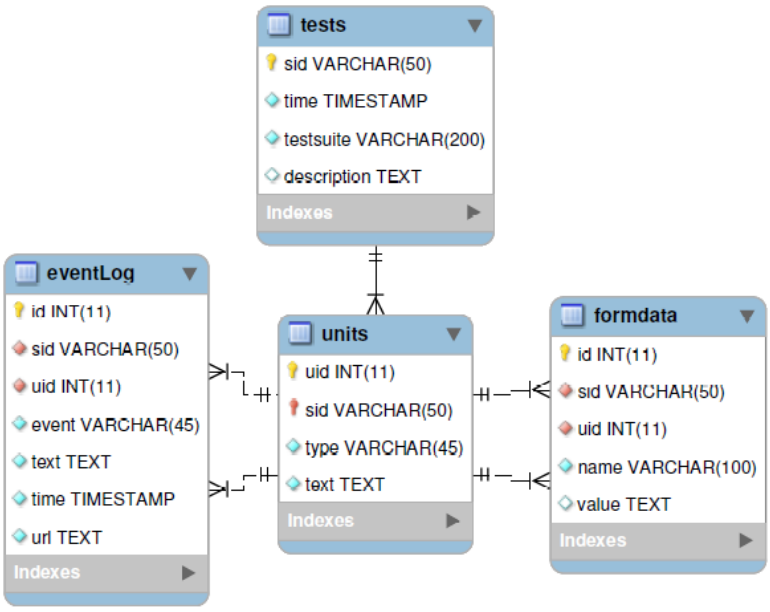

Abbildung 5.2: Übersicht der Datenbankstruktur

5.2.1 Tabelle tests

Die Tabelle *tests* (vgl. Abb. 5.3) ist die Haupttabelle der Datenbank. Sie enthält für jeden Test, den ein Proband macht, eine Zeile. Alle Einträge in den weiteren Tabellen können somit über diese Tabelle eindeutig zu einem bestimmten Testdurchlauf zugeordnet werden. Als Primärschlüssel besitzt die Tabelle das Feld *sid*. Dieses beherbergt die von PHP erzeugte Session-ID, eine eindeutige Zeichenfolge, die einen Besucher der Webanwendung identifiziert. Darüber hinaus gibt es noch die Felder *time*, *testsuite* und *description*. Die Spalte *time* enthält, wie der Name schon andeutet, den zur Eintragungszeit aktuellen Zeitstempel, der automatisch von MySQL eingetragen wird. *testsuite* und *description* sind von der ausgewählten Testsuite abhängig. Während *testsuite* schlicht den Dateinamen jener speichert, wird der Wert von *description*, falls vorhanden, aus der XML-Datei extrahiert und dient zur leichteren späteren Identifikation der Testsuite.

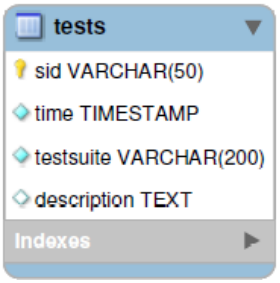

Abbildung 5.3: Tabelle tests

5.2.2 Tabelle units

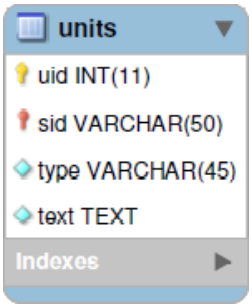

Abbildung 5.4: Tabelle units

Die Tabelle *units* (vgl. Abb. 5.4) dient, wie die Tabelle *tests*, Verwaltungszwecken. Alle Elemente, aus denen eine Testsuite besteht, – sämtliche Formulare und Aufgaben – werden in ihr gespeichert. So ist eine spätere Zuordnung von einzelnen erhobenen Daten zu der Aufgabe beziehungsweise zu dem Formular möglich. Selber besitzt die Tabelle einen Fremdschlüssel namens *sid*, die das gleichnamige Feld in der Tabelle *tests* referenziert. Somit ist auch eine Zuordnung zu dem jeweiligen Testlauf gegeben. Gleichzeitig ist *sid* ein Teil des zusammengesetzten Primärschlüssels, der auch das Feld *uid* umfasst. *uid* ist eine Zahl, welche die Reihenfolge der Elemente in der Testsuite repräsentiert. Darüber hinaus besitzt die Tabelle die Felder *type* und *text*, welche Informationen über die Art des Testelements enthalten. So gibt *type* den Typ an, also ob es sich um ein Formular („form“) oder eine Aufgabe („task“) handelt. Sollte das Element ein Formular sein, speichert *text* den Namen des Formulars. Bei einer Aufgabe enthält es die Aufgabenstellung.

5.2.3 Tabelle formdata

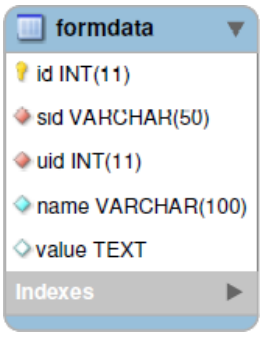

Abbildung 5.5: Tabelle formdata

Die dritte Tabelle der Datenbank heißt *formdata*. Sie wird zur Speicherung aller Daten, die aus Formularen stammen, genutzt. Dazu benutzt sie zum einen das Feld *name*. In ihm werden die Namen der Eingabefelder abgelegt. Zum anderen gibt es das Feld *value*, um den dazugehörigen Wert zu speichern. Um die Daten später wieder zuordnen zu können, müssen sie mit ihren zugehörigen Formularen verknüpft werden. Aus diesem Grund stellt der zusammengesetzte Fremdschlüssel aus den Feldern *sid* und *uid* eine Verbindung mit der Tabelle *units* her. Als Primärschlüssel nutzt die Tabelle *formdata* das Feld *id*. Als solches ist der Primärschlüssel ein von MySQL automatisch hochzählender nummerischer Wert.

5.2.4 Tabelle eventLog

Die Tabelle *eventLog* ist die letzte Tabelle, welche die Datenbank beinhaltet. Sie bildet den Kern der Protokollierung und sichert somit sämtliche Nutzeraktionen, die während eines Testes anfallen. Das Schema der Schlüssel ist identisch zu der Tabelle *formdata* aufgebaut. Dies bedeutet, dass es einen Fremdschlüssel bestehend aus *sid* und *uid* und einen Primärschlüssel *id* gibt. Die eigentliche Speicherung der Aktionen findet in den vier Feldern *event*, *text*, *time* und *url* statt. Das Feld *event* beinhaltet den Typ des Ereignisses. Für die möglichen Ereignisse sei auf das Kapitel 3.4 verwiesen. Daneben beherbergen die Felder *url* und *text* weitere Details zu dem Ereignis, und zwar zum einen auf welcher Webseite das Ereignis aufgetreten ist und zum anderen einen beschreibenden Text, der weitere Erläuterungen liefert. Ein aktueller Zeitstempel wird automatisch von MySQL im Feld *time* hinterlegt.

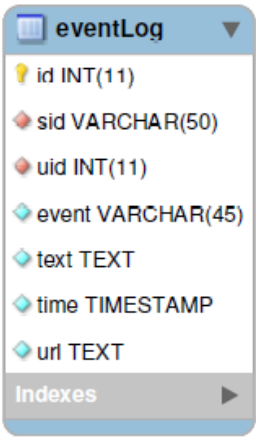

Abbildung 5.6: Tabelle eventLog

Kapitel 6

Implementierung – Webanwendung

In diesem Kapitel wird auf die Umsetzung der Webanwendung eingegangen. Es werden die einzelnen Bestandteile der Webanwendung vorgestellt und mithilfe von Diagrammen der Programmablauf verdeutlicht. Begonnen wird mit generellen Anmerkungen zu der Implementierung. Danach wird das Einlesen der Konfiguration der Testsuite erläutert, gefolgt von einem Überblick über die verschiedenen Oberflächen zur Darstellung der Testumgebung. Abschließend wird dann die Protokollierung betrachtet. An dieser Stelle sei nochmal das Kapitel 4 genannt, in dem schon einmal auf die grobe Aufteilung der Implementierung eingegangen wurde. Die Konfiguration wird mittels XML-Dateien vorgenommen, während die Datenhaltung der Messdaten per Datenbank geschieht. Der Quellcode ist auf Basis des MVC-Models strukturiert.

6.1 Allgemeines

Dieser Abschnitt beschäftigt sich mit generellen Anmerkungen, die die Implementierung der Testumgebung betreffen.

Ein Kernpunkt dieser ist das Management des aktuellen Teils des Testes, der dem Proband präsentiert wird. Dieser sollte auch bestehen bleiben, wenn beispielsweise die Seite neu geladen wird und sich nur verändern, wenn der Proband dies will. Aus diesem Grund wird der aktuelle Teil, nummeriert anhand seiner Position in der XML-Datei, in einer Session-Variable gespeichert und beim Laden der Seite so immer wieder identisch hergestellt. Mittels des GET-Parameter *next* kann der Inhalt dieser Variable verändert werden und damit auch, was dem Probanden angezeigt wird. Der Inhalt muss der Nummer des aktuellen Testteils inkrementiert um eins entsprechen, wenn also momentan Teil fünf angezeigt wird, muss *next* den Wert sechs enthalten. Alle anderen Werte sind unzulässig. *next* ist somit auch nur in der Lage, den direkt folgenden Teil auszuwählen, eine freie Auswahl ist nicht möglich. Gestartet wird eine Testsuite indem der GET-Parameter

Name	Funktion
instrument.php	Serverseitige Logik der Testumgebung
logger.php	Eintragen von Ereignissen in die Datenbank
instrument.js	Clientseitige Logik der Testumgebung
instrument.tpl	Darstellung der Testumgebung
dbzugang.php	Verbindung zur Datenbank
smarty_connect.php	Verwaltung von Smarty

Tabelle 6.1: Wichtige Dateien

testsuite mit dem Dateinamen, der die Testsuite beinhaltenden XML-Datei, übergeben wird.

Generell nutzt die Webanwendung eine PHP-Session, um einen Testdurchlauf anhand der Session-ID zu identifizieren. Auch werden in der Session Hilfswerte zur korrekten Steuerung der Testumgebung abgelegt. Um den Testdurchlauf zu beenden und die momentane Session zu löschen, ist der GET-Parameter *clear* vorgesehen, welcher keinen Wert braucht.

Eine Liste mit den wichtigsten Dateien und ihrer Funktionen ist in Tabelle 6.1 zu finden. Auf einzelne Dateien wird aber im Folgenden nicht näher eingegangen, sondern nur auf die generelle Umsetzung.

6.2 XML-Verarbeitung

Bei jedem Testdurchlauf wird eine XML-Datei und damit verbunden die darin enthaltene Testsuite abgearbeitet. Das Format der XML-Datei wurde bereits in Abschnitt 5.1 ausführlich beschrieben. Im Folgenden wird die programmseitige Verarbeitung dieser vermittelt.

Als erster Schritt muss natürlich eine XML-Datei bereitgestellt werden, aus der eingelesen wird. Dies geschieht aber in der Regel im Vorfeld des Testes durch das Personal und wird als gegeben vorausgesetzt. Sollte zu Testbeginn noch keine Datei ausgewählt worden sein, erscheint ein Dialog mit einer Auswahlliste aller verfügbaren Dateien. Beim Start eines neuen Testlaufs wird zunächst das *description*-Attribut des Elements *testsuite* ausgelesen und ein Eintrag in der Datenbank mit diesem Wert und anderen eingefügt (vgl. Kap. 5.2.1). Danach wird die Datei linear für jedes Element auf der ersten Ebene, das heißt für jedes Element direkt unterhalb des *testsuite*-Elements, durchlaufen. Jedes dieser Elemente wird über eine eindeutige Nummer, entsprechend der Position, identifiziert. Für jedes Element wird, sobald es an der Reihe ist, ein entsprechender Eintrag in die Datenbank geschrieben (vgl. Kap. 5.2.2). Das weitere Verhalten ist abhängig davon,

ob es sich bei dem aktuellen Element um eine Aufgabe oder ein Formular handelt.

Bei einem Formular werden zunächst die Attribute des Formulars ausgelesen und für eine weitere Verarbeitung gespeichert. Dann läuft eine Schleife über jedes Unterelement. Anhand des Typs werden für jedes Unterelement die benötigten weiteren Werte aus der XML-Datei gelesen und in einem Array hinterlegt. Sobald die Schleife durchgelaufen ist, existiert somit ein Array, welches alle Informationen über das Formular beinhaltet. Jetzt muss dieses nur noch an Smarty zur Darstellung übergeben werden.

Sollte es sich bei dem aktuellen Element nicht um ein Formular sondern um eine Aufgabe handeln, sind die benötigten Schritte einfacher. Eine Aufgabe hat keinen dynamischen Anteil bei der Konfiguration und somit müssen nur alle die Aufgabe spezifizierenden Unterelemente und Attribute eingelesen und als Werte an Smarty übergeben werden.

6.3 Oberflächen

Die Oberfläche der Webanwendung bildet die einzige Schnittstelle zwischen Mensch und Maschine. Somit muss diese für den Probanden einfach zu handhaben sein. Wenn der Proband sich bereits mit der Testumgebung stark auseinandersetzen muss, wird er zum Testbeginn eine andere Grundeinstellung haben, als wenn die Testumgebung intuitiv zu benutzen ist.

Die Testumgebung besteht aus insgesamt zwei verschiedenen Arten von Oberflächen, die der Proband zu sehen bekommt. Zum einen sind dies die Formulare und zum anderen die Aufgaben. Im Folgenden wird nun auf beide Arten getrennt eingegangen.

6.3.1 Aufgabenbearbeitung

Die Oberfläche, die während der Bearbeitung einer Aufgabe dargestellt wird, ist in Abbildung 6.1 zu sehen. Im oberen Teil der Webseite ist ein fester Bereich, der die komplette Steuerung der Testumgebung beinhaltet.

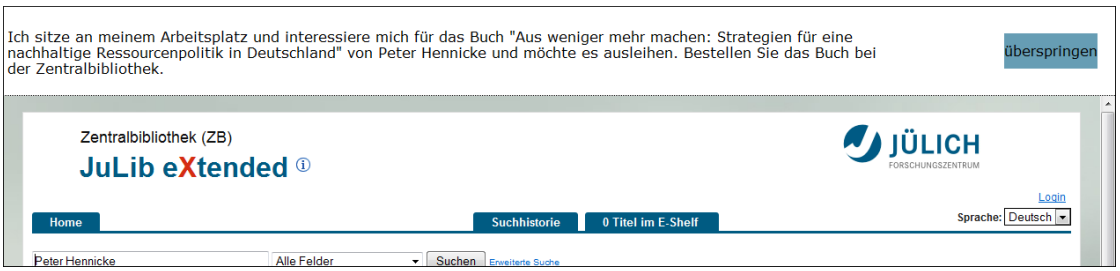

Abbildung 6.1: Aufgaben-Oberfläche (Ausschnitt)

Dieser Bereich ist dort permanent zu sehen, das heißt er scrollt nicht mit dem restlichen Inhalt. Im restlichen Fenster ist die zu testende Webseite dargestellt. Wenn man eine neue Aufgabe lädt, sind oben rechts zunächst zwei Buttons zu sehen. Nach einem Klick auf den Button „Start“ beginnt die eigentliche Aufgabe und im unteren Teil wird die entsprechende Webseite geladen. Bis dahin zeigt der untere Bereich eine weiße Seite an. Mit dem Button „Überspringen“ lässt sich die aktuelle Aufgabe beenden, und es wird sofort mit dem nächsten Teil des Testes fortgefahren. Neben den beiden Buttons wird im oberen Bereich die aktuell zu bearbeitende Aufgabe angezeigt, um dem Probanden durchgehend eine Gedankenstütze zu bieten.

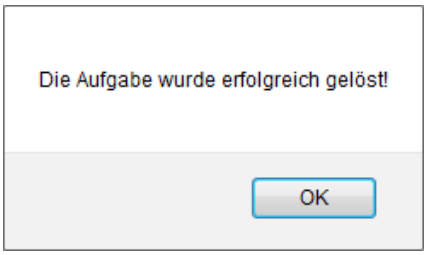

Abbildung 6.2: End-Dialog einer Aufgabe

Ist eine Aufgabe abgeschlossen, erscheint ein Dialog-Fenster (siehe Abb. 6.2), um den Probanden über das erfolgreiche Ende zu informieren, bevor der nächste Teil der Testsuite geladen wird.

6.3.2 Formular

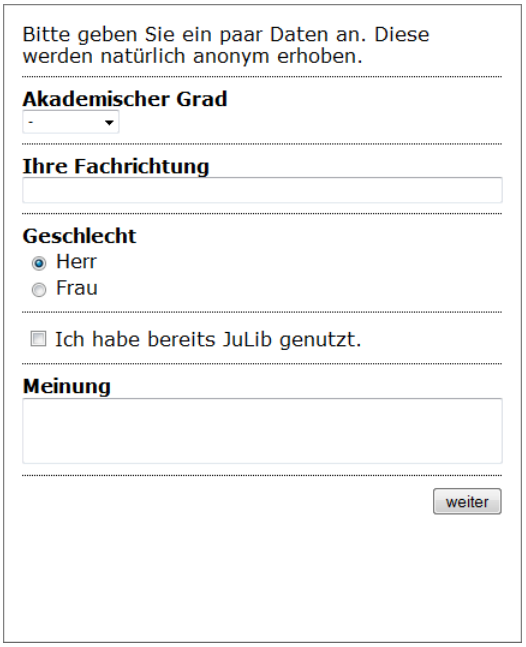

Abbildung 6.3: Beispiel-Formular

Die zweite Oberfläche, über welche die Webanwendung verfügt, ist für die Darstellung von Formularen zuständig. Im Gegensatz zu der Oberfläche für die Aufgabenbearbeitung muss hier ein dynamischer Inhalt angezeigt werden. Das Formular wird, wie in Abbildung 6.3 gezeigt, im Webbrowser dargestellt. Die Breite des weißen Bereichs ist dabei fest, während die Höhe sich dem Inhalt anpasst. Wie bereits in Kapitel 5.1.2 erläutert, wird der Inhalt des Formulars per XML definiert. Dabei gibt es keine Möglichkeit das Aussehen der einzelnen Elemente anzupassen. Dies geschieht direkt bei der Ausgabe des Formulars standardisiert, sodass ein einheitliches Design aller Formulare gewährleistet werden kann.

Soll ein Formular überspringbar sein, wird beim Laden des Formulars ein Dialog 6.4 mit entsprechendem Text angezeigt. Der Proband kann daraufhin wählen, ob er das Formular ausfüllen möchte oder nicht.

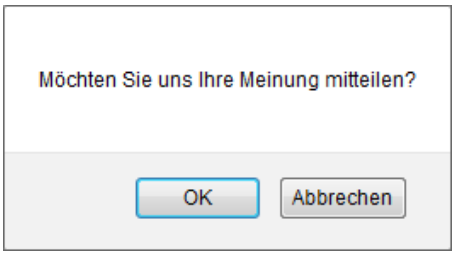

Abbildung 6.4: Überspringen-Dialog bei Formularen

6.4 Protokollierung

Die Protokollierung spielt eine wesentliche Rolle bei der Webanwendung, da sie zentraler Bestandteil des Verfahrens zur Usability-Evaluation ist. Auch die Umsetzung der Protokollierung in der Webanwendung ist, wie die Oberflächen, zweigeteilt (vgl. Abb. 6.5).

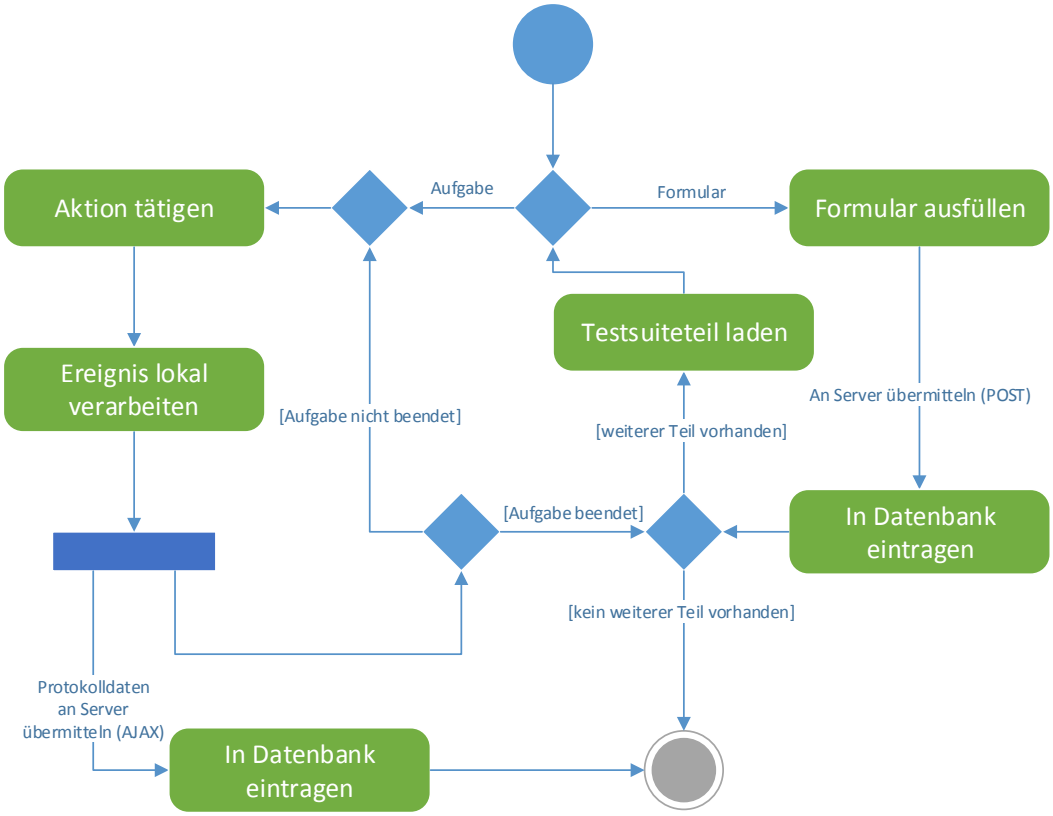

Abbildung 6.5: Aktivitätsdiagramm: Protokollierung

Die Formulardaten werden beim Abschicken als POST-Parameter an den Server geschickt, der diese dann ausliest und gemäß Kapitel 5.2 in die Datenbank überführt. Dabei

wird direkt der nächste Teil der Testsuite geladen und dem Proband angezeigt.

Im Gegensatz dazu kommt die Protokollierung der Nutzeraktionen nicht mit Standardmethoden von HTML bzw. PHP aus. Das Erfassen der Aktionen muss auf der Seite des Webbrowsers geschehen, da der Server von diesen nichts mitbekommen kann. Aus diesem Grund kommt an der Stelle JavaScript zum Einsatz. Mit JavaScript ist es möglich, auf eine ganze Reihe von Ereignissen zu reagieren, dazu zählen auch Maus- und Tastatureingaben. Um das Abfangen von den Eingaben näher zu erläutern, wird nun kurz das Beobachter-Entwurfsmuster[1] von JavaScript vorgestellt.

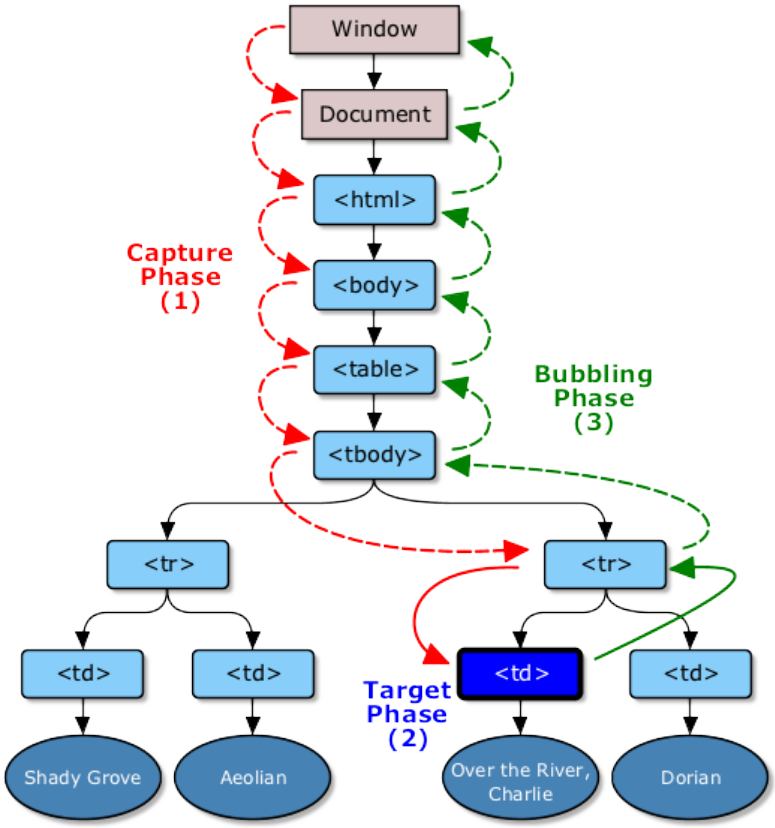

Abbildung 6.6: Ereignis-Weitergabe[HKL+13]

Auf ein Ereignis kann mit einem sogenannten Listener reagiert werden[HKL+13]. Dies sind Methoden, die ausgeführt werden, falls ein bestimmtes Ereignis auftritt. Zu einem Listener gehört dabei immer ein festes Ereignis. Außerdem ist ein Listener einem Element im DOM zugewiesen. Für den Fall, dass auf verschiedenen Ebenen des DOM verschiedene Listener für das gleiche Ereignis vorhanden sind, ist folgendes Vorgehen wichtig.

[1] http://de.wikipedia.org/wiki/Beobachter_%28Entwurfsmuster%29

Die Abarbeitung eines Ereignisses erfolgt in drei Phasen: 1. Capturing Phase, 2. Target Phase und 3. Bubbling Phase (vgl. Abb. 6.6). Ein Ereignis wird zunächst vom höchsten Element des DOM bis zum Element, welches das Ereignis ausgelöst hat, durchgereicht (Capturing Phase). Danach wird das Ereignis von diesem Element bearbeitet (Target Phase), um anschließend wieder bis zum obersten Element zu wandern (Bubbling Phase). Ein Listener muss entweder zur ersten oder zur dritten Phase zugeordnet werden und alle entsprechenden Listener auf dem Weg des Elements werden der Reihe nach ausgelöst. Zudem kann die Weitergabe des Elements von jedem Listener unterbunden werden.

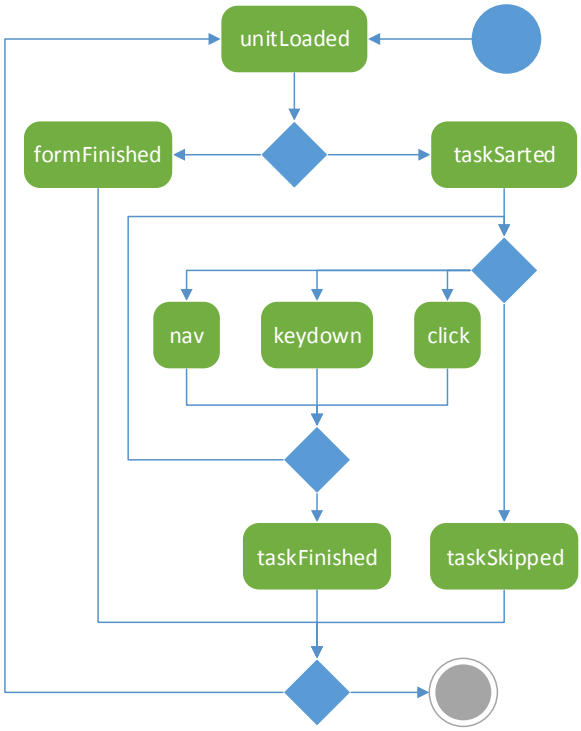

Abbildung 6.7: Ereignis-Abfolge

Bei der Protokollierung der Eingaben ist also die erste Phase das Mittel der Wahl. Die Listener müssen so nur noch im obersten Element des DOM registriert werden und bekommen alle Ereignisse der gesamten Webseite mit, da an dieser Stelle die Ereignisweitergabe nicht unterbunden werden kann. So kann kein Ereignis vor der Protokollierung durch die Listener verborgen werden.

Klickt nun zum Beispiel ein Proband auf ein beliebiges Element der zu testenden Webseite, wird ein entsprechendes Ereignis ausgelöst. Dieses wandert nun vom Fenster

und dessen Dokument, über das Body-Element bis zum Element, worauf geklickt wurde. Dort wird die eigentliche Aktion des Klickens ausgeführt, beispielsweise der Fokus auf dieses Element gesetzt. Danach wird das Ereignis wieder bis zum Fenster nach oben zurückgegeben. Da aber der Listener für Klick-Ereignisse im obersten Element des DOM registriert ist, wird dieser unmittelbar nach Auftreten des Ereignisses ausgelöst, noch bevor das Ereignis durch den DOM wandert.

Es wird ein Listener für die Aufzeichnung der Mausklicks und zwei Listener für Tastatureingaben benötigt, wovon einer auf das Herunterdrücken einer Taste und der andere auf das Loslassen einer Taste reagiert. Der zuletzt genannte Listener hat keine außerordentliche Bewandtnis, sondern ist nur dafür zuständig, den Zustand der Shift-Taste richtig zu erkennen. Zusätzlich wird auf dem IFrame ein Listener registriert, der auf Seitenwechsel reagiert. Die Listener schicken unmittelbar nach Auftreten des entsprechenden Ereignisses mittels AJAX im Hintergrund das Ereignis mit zusätzlichen Informationen an den Server. Die zusätzlichen Informationen können dem Kapitel 3.4 entnommen werden.

Auf der Serverseite müssen nur noch die übergebenen Werte ausgelesen und in die Datenbank eingetragen werden.

In Abbildung 6.7 ist nochmals ersichtlich, in welcher Reihenfolge die protokollierten Ereignisse auftreten und folglich auch in der Datenbank vorkommen. Zunächst wird immer das reine Laden des aktuellen Testabschnitts mit dem Ereignis *unitLoaded* protokolliert. Handelt es sich um ein Formular, wird nur noch das Abschicken des Formulars mit *formFinished* registriert. Bei einer Aufgabe allerdings fallen mehrere Ereignisse während der Bearbeitung an. Das Ereignis *taskStarted* wird durch den Klick auf den Start-Button ausgelöst, und ab diesem fängt die eigentliche Bearbeitung an. Während dieser können in Abhängigkeit von den Aktionen des Probanden die Ereignisse *click*, *keydown* und *nav* auftreten. Abgeschlossen wird eine Aufgabe schließlich entweder mit *taskSkipped*, wenn die Aufgabe übersprungen wird, oder mit *taskFinished*, falls die Aufgabe erfolgreich beendet wird.

Bei der kompletten Protokollierung muss allerdings darauf geachtet werden, dass keine personenbezogenen Daten, wie Nutzernamen oder Passwörter, erfasst werden. Um eine anonyme Datenerfassung zu gewährleisten, ist es möglich eine Liste von HTML-Elementen zu definieren, die von der Protokollierung ausgenommen sind. Diese wird zwar serverseitig festgelegt, aber clientseitig umgesetzt. So prüft die entsprechende JavaScript-Funktion vor der Übermittlung der Daten an den Server, ob das Element in der Liste steht und unterlässt gegebenenfalls das Senden. Auf diese Weise werden keine personenbezogenen Daten protokolliert.

Kapitel 7

Fazit und Ausblick

Dieses Kapitel rekapituliert die Arbeit und fasst die Inhalte nochmals zusammen. Zusätzlich wird ein Ausblick gegeben.

7.1 Zusammenfassung und Fazit

In dieser Arbeit wurde ein Verfahren für eine Usability-Evaluation in der Zentralbibliothek entwickelt und eine darauf basierende Evaluations-Umgebung entworfen.

Es wurde ein empirisches und summatives Verfahren gewählt, welches mithilfe von Probanden durchgeführt werden soll. Als Zielgruppe wurden Wissenschaftler, Doktoranden sowie Studenten gewählt, da diese am meisten mit fachlicher Literatur in Kontakt kommen. Als Probanden sollen Nutzer des Lesesaals sowie Teilnehmer der zahlreichen Schulungen der Zentralbibliothek gewonnen werden. Den Probanden werden Aufgaben gestellt, die diese dann bearbeiten sollen. Dabei wird protokolliert, welche Aktionen sie getätigt haben, um daraus Erkenntnisse über die Nutzbarkeit der Seite zu erlangen. Neben den Aufgaben ist es auch möglich, den Probanden Formulare zum Ausfüllen zu geben, damit neben den quantitativen Daten der Aufgaben auch eine Reihe qualitativer Daten gesammelt werden kann. Die Formulare sind nicht fest vorgegeben, sondern können dynamisch für jeden Test einzeln erstellt werden.

Alle erhobenen Daten werden übersichtlich in einer Datenbank gespeichert, um sie für eine spätere Auswertung auf einfache Weise abrufen zu können. Die Konfiguration einer Testsuite wird mittels XML-Datei vorgenommen, damit die Erstellung von neuen Testsuites auch durch nicht-fachkundiges Personal durchgeführt werden kann. Außerdem bietet XML durch seine Struktur eine sehr gute Übersicht sowie Erweiterbarkeit.

Die Evaluations-Umgebung kann nun eingesetzt werden, um eine Usability-Evaluation durchzuführen. Nebenbei wurde bei der Implementation darauf geachtet, eine möglichst große Anpassbarkeit zu gewährleisten. So kann die Umgebung nicht nur für JuLib, sondern auch für die Evaluation weiterer Webanwendungen eingesetzt werden.

Insgesamt kann festgestellt werden, dass alle seitens der Zentralbibliothek gestellten Anforderungen erfüllt wurden. Genauer sind dies: Protokollierung der Aktionen, dynamische Formulare, keine Erhebung personenbezogener Daten sowie keine Veränderungen am lokalen PC.

7.2 Ausblick

Im Anschluss dieser Arbeit sind einige mögliche Fortsetzungen denkbar. Zum einen gäbe es Erweiterungsmöglichkeiten die Webanwendung betreffend und zum anderen wurden in dieser Arbeit noch nicht die Auswertung der Messdaten sowie der praktische Einsatz durchgeführt.

Vor allem der praktische Einsatz stellt den nächsten Schritt dar. Dieser könnte Verbesserungsmöglichkeiten in der Webanwendung und Unzulänglichkeiten in der Erhebung der Messdaten aufzeigen. Auch liefert ein erster praktischer Versuch Messdaten, mithilfe derer ein Schema zur systematischen Auswertung erarbeitet werden kann.

Die Auswertung könnte im ersten Schritt eine Übersicht der protokollierten Ereignisse umfassen, aus der schnell erkenntlich wird, was der Proband gemacht hat. Außerdem könnten somit statistische Werte über Verweildauern in einzelnen Testabschnitten und ähnliches erstellt werden. Auf Grundlage der Auswertung können dann wiederum Rückschlüsse auf die Vollständigkeit der Erhebung getätigt werden, um gegebenenfalls weitere Messdaten einzuführen.

Aber auch an der Webanwendung selber könnten noch Erweiterungen vorgenommen werden. So ist es mit der momentanen Implementation nicht möglich, externe Webseiten zu protokollieren. Ein weiterer Punkt für eine Verbesserung bildet die Erkennung des Aufgabenendes. Diese könnte vom puren Vergleich der aktuellen URL mit einer gegebenen URL zu einem ereignisabhängigen Verfahren geändert werden. So könnte beispielsweise auch die Aufgabe mit einem Mausklick auf ein bestimmtes Element beendet werden.

Abschließend lässt sich sagen, dass mit dieser Arbeit der Grundstein für eine zukünftige Usability-Evaluation in der Zentralbibliothek gelegt wurde.

Anhang A

Begriffserklärung

Architekturmuster Grundlegende Organisation der Komponenten einer Anwendung.

Dokumenttypdefinition Satz an Regeln, um Typen von Dokumenten zu deklarieren.

Ereignis (Informatik) Begebenheit, die eine Aktion und ggf. eine Zustandsänderung herbeiführt.

Evaluation Methodische Erhebung von Daten zur Bewertung eines Sachverhalts.

Eye-Tracking Aufzeichnung der Augenbewegungen.

Heatmap Diagramm zur Visualisierung von Daten, dessen Werte als Farbe in einer zweidimensionalen Fläche dargestellt werden.

Heuristik Analytisches Vorgehen, um mit begrenztem Wissen eine Aussage über einen Sachverhalt zu treffen.

IFrame HTML-Element zur Einbindung anderer Webinhalte in das eigene Dokument.

Proxy Vermittler, der Anfragen entgegennimmt, ggf. verarbeitet und weiterleitet.

Same-Origin-Policy Sicherheitskonzept, das den Zugriff auf Objekte einer anderen Quelle unterbindet.

Session Stehende Verbindung zwischen Client und Server. Der Client ist dabei eindeutig identifizierbar durch den Server.

Severity-Rating Einordnung von Problemen in Kategorien.

Testsuite	Sammlung von einzelnen Tests.
Usability	Gebrauchstauglichkeit von einem Produkt.
Webapplikation	Programm, welches beim Client im Webbrowser läuft. Sie werden meist auf einem Webserver gespeichert und von diesem an die Clienten verteilt.

Literaturverzeichnis

[AWS06] Atterer, Richard ; Wnuk, Monika ; Schmidt, Albrecht: Knowing the User's Every Move: User Activity Tracking for Website Usability Evaluation and Implicit Interaction. In: *Proceedings of the 15th International Conference on World Wide Web*. New York, NY, USA : ACM, 2006 (WWW '06). – ISBN 1–59593–323–9, 203–212

[Dan07] DanielSHaischt: *Prozessfluss AJAX*. Wikipedia. `http://de.wikipedia.org/wiki/Ajax_%28Programmierung%29#mediaviewer/Datei:Prozessfluss-ajax.svg`. Version: Juni 2007. – Stand: 07.07.2014

[Heg03] Hegner, Marcus: *IZ-Arbeitsbericht Nr. 29, Methoden zur Evaluation von Software*. Informationszentrum Sozialwissenschaften der Arbeitsgemeinschaft Sozialwissenschaftlicher Institute e.V. (ASI), 2003 `http://www.gesis.org/fileadmin/upload/forschung/publikationen/gesis_reihen/iz_arbeitsberichte/ab_29.pdf`

[HKL+13] Hégaret, Philippe L. ; Kacmarcik, Gary ; Leithead, Travis ; Pixley, Tom ; Höhrmann, Björn ; Schepers, Doug ; Rossi, Jacob: Document Object Model (DOM) Level 3 Events Specification / W3C. 2013. – W3C Working Draft. – http://www.w3.org/TR/2013/WD-DOM-Level-3-Events-20131105/

[ISO99] *Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten - Teil 11: Anforderungen an die Gebrauchstauglichkeit*. Beuth Verlag, 1999 (DIN EN ISO 9241-11)

[Luc11] Luca, Helena: *Usability-Studie zu KonSearch : Evaluation der neuen Literatursuchmaschine der Universität Konstanz*. `http://nbn-resolving.de/urn:nbn:de:bsz:352-168427`. Version: 2011

[OHE01] Oertel, Karina ; Hein, Oliver ; Elsner, Antje: The RealEYES project - usability evaluation with eye tracking data. In: Hirose, Michitake (Hrsg.):

Proceeding of 8th International Conference on Human-Computer Interactions (INTERACT 2001). Dept. of Comput. Sci., Rostock Univ., Germany : IOS Press, 2001

[Pen06] Penzo, Matteo: *Evaluating the Usability of Search Forms Using Eyetracking: A Practical Approach*. http://www.uxmatters.com/mt/archives/2006/01/evaluating-the-usability-of-search-forms-using-eyetracking-a-practical-app php. Version: January 2006. – Stand: 16.07.2014

[Reg10] RegisFrey: *MVC-Process*. Wikipedia. http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller#mediaviewer/File:MVC-Process.svg. Version: Mai 2010. – Stand: 07.07.2014

[TU] TU Berlin: *Lautes Denken - Usability-Wiki*. http://www.uselab.tu-berlin.de/wiki/index.php/Lautes_Denken. – Stand: 07.07.2014

[Umf] *Umfrage, Usability-Fragebogen, Nutzerbefragung, schriftliche Befragung, Online-Fragebogen - usability-toolkit.de*. http://usability-toolkit.de/usability/usability-methoden/umfrage/. – Stand: 14.07.2014

[Wik] *Definition Usability-Test*. http://de.wikipedia.org/wiki/Usability-Test. – Stand: 07.07.2014

[Wil14] Wilson, Chauncey: Chapter 3 - Perspective-Based {UI} Inspection. Version: 2014. http://dx.doi.org/http://dx.doi.org/10.1016/B978-0-12-410391-7.00003-8. In: Wilson, Chauncey (Hrsg.): *User Interface Inspection Methods*. Boston : Morgan Kaufmann, 2014. – DOI http://dx.doi.org/10.1016/B978–0–12–410391–7.00003–8. – ISBN 978–0–12–410391–7

[WtB11] Weinhold, Thomas ; Öttl, Sonja ; Bekavac, Bernard: BibEval - Ein webbasierter Kriterienkatalog zur Usability-Evaluation von Bibliothekswebsites. In: *Information Wissenschaft&Praxis* 1 (2011), S. 11–16

Anhang B

XML-Konfiguration

Listing B.1: testsuite.dtd

```
<!ELEMENT testsuite (form, task)*>

<!ELEMENT form (text, input, submit)*>
<!ELEMENT task (start, end, text)>

<!ELEMENT text (#PCDATA)>
<!ELEMENT input ((option)*|#PCDATA|EMPTY)>
<!ELEMENT submit (#PCDATA)>

<!ELEMENT start (#PCDATA)>
<!ELEMENT end   (#PCDATA)>

<!ATTLIST testsuite
   blacklist CDATA #IMPLIED
   description CDATA #IMPLIED
>

<!ATTLIST form
   name CDATA #REQUIRED
   prompt (true|false) #IMPLIED
>

<!ATTLIST text
   label CDATA #IMPLIED
>
```

```
<!ATTLIST input
   label      CDATA #IMPLIED
   type       (text|checkbox|radio|textarea|select) #REQUIRED
   name       CDATA #IMPLIED
   checked    (true|false) #IMPLIED
>
```